Utilização de sistemas de CAM para programação de máquinas-ferramenta

António Luís Galamba de Oliveira Felgueiras de Carvalho

Dissertação do MIEM – Opção de Automação

Orientadores:

Prof. Manuel Romano dos Santos Pinto Barbosa

Prof. Paulo Augusto Ferreira de Abreu

Faculdade de Engenharia da Universidade do Porto

Mestrado Integrado em Engenharia Mecânica

Julho 2013

Resumo

Esta dissertação aborda os aspetos inerentes à utilização de sistemas CAM na programação de máquinas-ferramenta de uma forma geral, e em particular para a máquina-ferramenta Okuma MC-40VA, com o *software* Mastercam X5, procurando responder a duas questões importantes: o que implica usar um sistema CAM, e o que afeta a qualidade da geração de código de comando numérico?

O trabalho apresentado começa por uma revisão da evolução do CAM como aplicação e como conceito. Faz depois uma exposição do funcionamento e estrutura do Mastercam X5, ilustrando os conceitos revistos através de três exemplos práticos. De seguida faz um exemplo de configuração de uma máquina virtual representativa da Okuma MC-40VA, discriminando todas as suas fases, desde a recolha de informação, até à edição de um pós-processador. Os resultados e a sua discussão consistem na avaliação da qualidade das funcionalidades do Mastercam X5, assim como do pós-processador editado.

Conclui-se com esta dissertação que o contributo do utilizador no uso de um *software* CAM ainda é significativo, essencialmente do tipo técnico, e a qualidade da geração de código de comando numérico depende essencialmente da qualidade dos percursos gerados.

CAM based machine tool programming

Abstract

This thesis is a study on what aspects are associated to machine tool programming through usage of CAM software. It begins to do so in a broad sense of the term CAM, and then narrows it into the commercial software Mastercam X5, and the machine tool Okuma MC-40VA. It seeks to answer two questions: what implications are there in using CAM software and what influences the quality of the output numerical control code?

It starts by reviewing the evolution of CAM both as software and as a concept. Then it displays its features and software usage structure, showing reviewed concepts through three practical cases. After that it performs an exercise that consists in configuring a virtual machine as a representation of the Okuma MC-40VA machine tool, reporting all steps, from machine related information gathering, up to editing a post-processor. Final results and their discussing include the evaluation of both Mastercam X5's features and the edited post-processor.

This thesis' conclusions find that there is still significant end-user technical intervention in using CAM software to generate numerical control code, and its quality will essentially depend on the quality of proposed machining paths.

Agradecimentos,

Aos orientadores,
Professor Manuel Romano dos Santos Pinto Barbosa,
Professor Paulo Augusto Ferreira de Abreu,
Pelo apoio, disponibilidade e paciência.

Ao professor Francisco Teixeira de Freitas,
Pela organização da, e dedicação à disciplina de dissertação.

Ao professor Joaquim Fonseca,
Pela ajuda e disponibilidade.
Ao corpo técnico das oficinas de mecânica da FEUP,
Pela ajuda e disponibilidade.

Aos professores Francisco Xavier de Carvalho e Francisco Pires,
Pela estimada amizade.

À minha família
Pelo inesgotável apoio.

Rita,
Obrigado.

Aos meus amigos.

Índice

1 – Introdução 7
1.1 – Motivação 7
1.2 – Objetivos 9
2 – Produção Assistida por Computador 11
2.1 – Contextualização dos sistemas CAM 11
2.2 – CAM e tecnologia CN 13
2.3 – Apoio à tomada de decisão e planeamento tecnológico 15
2.4 – Aplicações industriais 16
2.5 – Estrutura de software 16
2.6 – Mercado CAM 17
3 – Software de CAM: Mastercam X5 23
3.1 – Interface 24
3.2 – Funcionalidades 25
3.3 – Estrutura do funcionamento do Mastercam X5 29
4 – Exemplos de aplicação do Mastercam X5 41
4.1 – Modelo 1 – Operações de contorno 2D 42
4.2 – Modelo 2 – Operações 3D com superfícies complexas 62
4.3 – Modelo 3 – Operações 3D com orientação radial 72
5 – Configuração do Mastercam para a máquina Okuma MC-40VA 77
5.1 – Criação da máquina virtual 77
5.2 – Configuração de um pós-processador 79
5.2.1 – Estrutura de um pós-processador no Mastercam X5 80
5.2.2 - Referências do ficheiro NCI 83
5.2.3 – Requisitos operacionais da implementação de um pós-processador 84
5.3 – Características da máquina Okuma MC-40VA 86
5.4 – Implementação da máquina virtual 97
5.5 – Edição de pós-processadores 110
6 – Avaliação da utilização do Mastercam 113
6.1 – Avaliação das funcionalidades do Mastercam X5 114

6.2 – Avaliação do código de CN gerado 122
6.3 – Comentário à avaliação das funcionalidades do Mastercam X5 128
6.4 – Comentário à edição de pós-processadores 130
7 – Conclusões e proposta de trabalhos futuros 132
Bibliografia.....136
ANEXO A : PROGRAMAS DE CÓDIGO DE CN GERADOS EM 6.2.....138
ANEXO B : PROGRAMAS DE CÓDIGO DE CN GERADOS EM 6.2.....140
ANEXO C : EXCERTOS DO PÓS-PROCESSADOR MODIFICADO.....142
ANEXO C : PÓS-PROCESSADOR MODIFICADO, NA ÍNTEGRA 158

1 – Introdução

O uso de sistemas CAM tem provocado, na indústria utilizadora de máquinas ferramenta de controlo numérico computorizado (CNC), crescente interesse, desde pouco tempo depois do seu nascimento, até aos dias de hoje.

Os benefícios proporcionados pela utilização de CAM, especialmente no respeitante ao aumento da produtividade e redução dos custos, colocaram esta tecnologia no foco da investigação.

Esta dissertação abordará esta utilização, analisando em que ponto se encontra a automatização do CAM.

Neste capítulo será feita uma introdução ao tema, mostrando a sua motivação e posteriormente os seus objetivos.

1.1 – Motivação

Constatou-se, nos Estados Unidos da América, após a Segunda Guerra Mundial, que esforços conduzidos para que a indústria fosse mais competitiva estavam concentrados quase exclusivamente na engenharia do produto, e não nos sistemas de produção [1]. Demonstrando o erro da forma de pensar norte-americana, a indústria japonesa, com grande sucesso, investia em sistemas de produção mais eficientes. Consequentemente estaria lançada uma mudança de paradigma [1].

Esta mudança de paradigma levou a que, nos EUA, no final da década de 1940, se dessem grandes inovações. A necessidade de se produzir peças com geometrias complexas, assim como se conseguir uma maior diversidade de formas numa mesma máquina, a rápida evolução e crescente competitividade dos mercados, e a aplicação de critérios de qualidade e eficiência mais exigentes, motivaram a criação dos primeiros sistemas de controlo digital. Desde aí que a utilização de sistemas de comando numérico (CN) começou a alterar a forma como a máquina-ferramenta é automatizada [1].

Os primeiros sistemas implementados tiveram reduzido aproveitamento. Na sua primeira aplicação – a indústria aeronáutica norte-americana – a adaptação a estes novos sistemas foi gradual e acidentada. Apenas melhorias no controlador do comando numérico e na interface controlador-máquina puderam garantir por fim o sucesso [1].

A introdução do comando numérico levou a que se exigissem novas competências aos utilizadores destes sistemas. Seria agora necessário programar as máquinas, o que, sendo executado de uma forma manual, era uma tarefa complexa, relativamente desconhecida, de elevado consumo de tempo, e com azo a erros [1].

Com o advento do computador foi possível acelerar este processo de programação e diminuir os erros a si associados. Um dos desenvolvimentos mais significativos consistiu no desenvolvimento (iniciado no ano de 1955), de uma linguagem de programação de alto nível (APT- *Automatically Programmed Tool*) para elaboração de

programas de comando numérico [1]. Tirando partido do computador, a linguagem APT disponibiliza códigos específicos para definição de formas geométricas e de parâmetros associados à maquinagem, resultando numa especificação de percursos de maquinagem num formato standard, intermédio, das sucessivas posições da peça e ferramenta. Este passo intermédio permitiu tornar a geração do código de CN mais geral, isto é independente de uma máquina em particular. No entanto obrigaria à utilização de um segundo programa (designado "pós-processador") para configurar o programa de acordo com características específicas de cada máquina [1]. Esta linguagem e respetivo pós-processamento podem assim ser considerados como representativos da forma inicial e mais elementar de CAM (*Computer Aided Manufacturing*).

Esta estrutura, processador genérico e processador específico, são ainda hoje usados na implementação dos modernos sistemas ou linguagens de programação gráfica para CN, i.e. as aplicações, ou *software* CAM atual [1].

Através da natural junção do CAM e dos sistemas CAD, deu-se uma gradual automatização da programação de rotinas de CN. Isto tornou essa mesma tarefa mais acessível não só aos utilizadores dos equipamentos de maquinagem CNC, mas também a engenheiros de projeto, fazendo com que nessa fase pudessem mais facilmente ser incluídas considerações relativas à produção. Passar-se-ia a designar esta integração como CAD/CAM [1].

O conceito de CAM evoluiu até aos dias-de-hoje. Atualmente está relacionado com tomada de decisões associadas ao planeamento tecnológico, i.e. "como fazer um produto?", incluindo processamento de informação e preparação de produção [2].

Desde a sua criação até à atualidade, estes sistemas – no contexto da automação industrial – estão naturalmente dependentes da capacidade de incorporação das tecnologias informáticas. A utilização de equipamentos de CNC é um bom exemplo dessa associação, em particular desde a fase de elaboração de um programa de comando numérico até à sua execução na máquina-ferramenta, para dar forma a um bloco de material [1].

Apesar desta evolução, ainda hoje a utilização desta tecnologia levanta questões relativas aos recursos humanos e técnicos a disponibilizar para que a sua aplicação seja bem-sucedida, bem como às suas capacidades [1] [3]. Este facto está intrinsecamente ligado ao grau de automatização permitido pelos sistemas CAM na elaboração de programas de CN a serem executados numa dada máquina, ou seja:

- Qual o contributo do utilizador neste processo?
- De que depende a qualidade dos programas gerados?

A motivação deste trabalho é o de responder a estas questões, e o de aferir, de uma forma geral para as ofertas de *software* CAM presentes no mercado atual e em particular para o *software* Mastercam X5, como se dá essa intervenção.

1.2 – Objetivos

Baseado na motivação explanada no ponto anterior e o objetivo global de avaliar o grau de participação do utilizador na elaboração de programas de CN usando uma aplicação de CAM atual (Mastercam), os objetivos principais para este trabalho consistem em:

- Avaliar as funcionalidades disponíveis neste tipo de aplicações;
- Avaliar a flexibilidade, ou possibilidades de configuração destas aplicações;
- Configurar esta aplicação para uma máquina em particular.

Para poder atingir estes objetivos deverá ser realizado:

- Teste de funcionalidade do Mastercam usando exemplos de aplicação de diferentes operações de maquinagem;
- Configuração de pós-processadores para uma máquina em particular;
- Criação de um modelo virtual de uma máquina para simulação de operações de maquinagem.

Com base no trabalho realizado, a escrita desta dissertação organizou-se nos capítulos seguintes para além desta introdução. No Cap. 2, aborda-se uma caracterização e revisão do conceito de Produção Assistida por Computador (CAM). Esta abordagem contempla a sua versão mais comum e limitada, como *software* específico para facilitar a elaboração de programas de Comando Numérico para máquinas-ferramenta, até uma mais geral, envolvendo todas as atividades numa empresa desde a fase de planeamento tecnológico até à gestão e controlo dos equipamentos na produção. Com base na contextualização inicial de sistemas de CAM apresentadas no Cap.2, no Cap. 3 começa-se por uma abordagem global do *software* Mastercam, versão X5 utilizada neste trabalho, de modo a introduzir a sua estrutura e funções principais, com ênfase nas capacidades de configuração de acordo com o interesse do utilizador e de identificar o grau de automação possível. A análise das funções principais é complementada no Cap. 4 com base na definição e exemplificação de peças-tipo e na elaboração dos respetivos programas de maquinagem. No Cap. 5, apresenta-se com detalhe o processo de configuração de uma máquina virtual representativa de uma máquina real (Okuma MC-VA40) e de um pós-processador. No Cap. 6 apresenta-se a avaliação feita às funcionalidades do Mastercam e ao desenvolvimento do pós-processador, assim como um comentário a ambos. Finalmente no Cap. 7 conclui-se o trabalho e fornecem-se indicações para possíveis trabalhos futuros.

2 – Produção Assistida por Computador

Mostrou-se, no capítulo anterior, o conceito de CAM como resposta a uma indústria em busca de maior eficiência e rigor.

Neste capítulo aborda-se o conceito de CAM num contexto amplo, e num contexto mais restrito, focando aspetos da atualidade do CAM, como o estado-da-arte e situação do mercado.

Na subsecção seguinte analisa-se o estado-da-arte, dividindo-o em cinco aspetos principais: acompanhamento das tecnologias de CN; apoio à tomada de decisão e planeamento tecnológico; aplicações industriais; estrutura de *software;* mercado CAM. Faz-se também uma breve revisão da evolução tecnológica sofrida até à última década.

2.1 – Contextualização dos sistemas CAM

A designação CAM – *Computer Aided Manufacturing* – pode em geral ser interpretada segundo duas perspetivas. Por um lado englobando, de uma forma mais abrangente, todos os sistemas baseados em computadores (i.e. *software* e hardware) para apoio às funções de planeamento tecnológico, tomada de decisões e controlo dos equipamentos na produção. Por outro lado, numa perspetiva mais limitada o CAM restringe-se às soluções comercialmente disponíveis para ajudar na elaboração de programas de CN, tendo por base uma linguagem de programação, que atualmente se distingue pelas suas capacidades gráficas. Esta segunda visão dos sistemas de CAM é naturalmente mais facilmente reconhecida devido às soluções disponíveis no mercado e à sua grande utilização em empresas que utilizam máquinas de CN. Apesar de aplicação mais restrita, estas soluções também se enquadram, e muitas vezes são, a base da perspetiva mais abrangente [1] [2]. Este trabalho centra-se nas soluções comerciais para elaborar programas de CN mas tendo em conta a sua inserção nas restantes atividades de conceção, planeamento e controlo da produção (Figura 1).

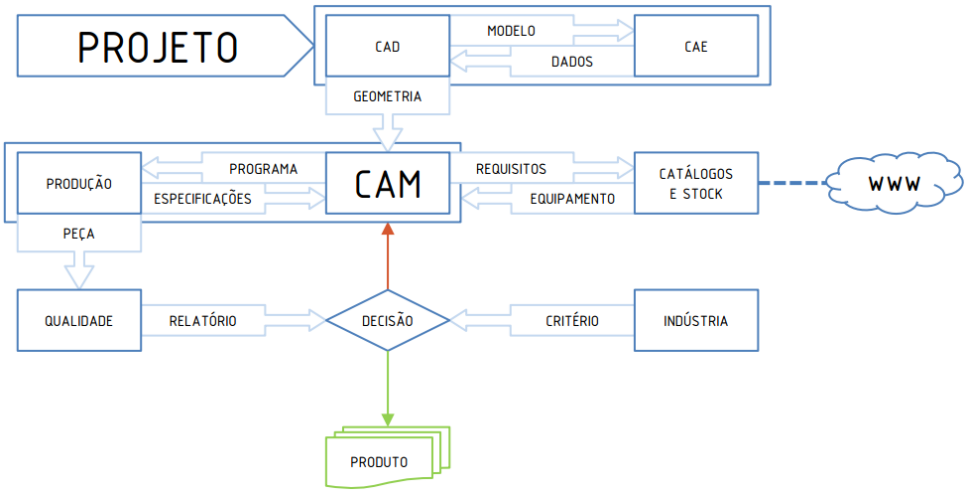


Figura 1 - Representação esquemática do CAM no contexto da engenharia.

Desta forma o CAM, na perspetiva mais prática, insere-se na atividade produtiva como um auxiliar da articulação entre a conceção e a produção. Os seus requisitos são o produto do trabalho de projeto – neste caso, uma geometria[1] – e informações técnicas relativas ao equipamento a utilizar, nomeadamente especificações de máquinas e ferramentas. Em todos os processos de maquinagem surgem questões como qual a melhor estratégia de maquinagem a adotar, e como obter o programa de comando numérico que permita a sua realização. Ainda nesta perspetiva, pode considerar-se que a funcionalidade final[2] do CAM é gerar o código de comando numérico para implementar essa estratégia numa máquina particular [1] [2] [4].

A justificação para o seu desenvolvimento e utilização deriva da crescente complexidade normalmente associada às peças a maquinar e ao tempo de planeamento necessário para escolher as melhores soluções. Por outro lado, sendo necessário um programa de CN para cada peça diferente, às funcionalidades destas soluções para rapidamente produzirem esse novo programa, está naturalmente associada à capacidade de reconfiguração do sistema de produção e consequentemente a sua flexibilidade. Minimizar o tempo despendido nestas funções, e também maximizar a sua eficiência, justifica a utilização e evolução contínua do CAM i.e. do suporte que proporciona à execução destas funções [1].

As funcionalidades principais de um sistema de CAM, nesta perspetiva de solução comercialmente disponível para gerar programas de CN, incluem [5] [6] [7]:

- Definição da geometria em bruto e maquinada (forma inicial e final);
- Escolha dos equipamentos de maquinagem;
- Escolha de ferramentas;
- Criação de percursos de maquinagem;
- Simulação das operações de maquinagem;
- Estimação dos tempos de maquinagem;
- Configuração para pós-processamento e geração de código de comando numérico.

Estas funcionalidades, desde a sua inclusão nos sistemas CAM, sofreram várias evoluções. Outras – mais modernas – seriam acrescentadas para fazer face a novas aplicações.

Na subsecção seguinte irá ser feita uma revisão que engloba as diversas evoluções que o CAM – na sua vertente comercial – tem vindo a sofrer.

[1] Embora se aplique aqui o conceito de geometria, ele tem neste contexto um significado mais amplo, ou seja de projeto, considerando também aspetos como tolerâncias e rugosidades.

[2] Entenda-se que aqui, funcionalidade final, entende-se como aquela que produz a "ponte" para a continuação do fluxo de trabalho.

2.2 – CAM e tecnologia CN

Originalmente, na década de 1970, as ofertas de *software* de CAM incluíam apenas funções essenciais da linguagem APT e do pós-processador, auxiliadas pela interface gráfica disponibilizada pelo CAD [1]. Durante este período, e até à adaptação do computador pessoal como suporte para programas CAM (na década de 1980), as ofertas consistiam em um só produto que englobava tanto *hardware* como o *software*. Estas ofertas eram naturalmente dispendiosas, mas pouco fiáveis. Nesta fase, o CAM começaria a suportar não só a perfilagem de chapa, mas também a fresagem e torneamento [1] [8]. A década de 1980 viu a maturação da implementação gráfica, e da sua melhor utilização, o que levou, no CAM, à criação de algoritmos de geração de percursos com seleção de geometria[3]. Empresas como a Delcam PLC e a CNC *Software* (responsável pelo *software* Mastercam) lançaram, nesta década, soluções comerciais já pensadas para computadores pessoais [5] [6]. Do ano de 1990 ao ano 2000, os maiores desenvolvimentos deram-se na tecnologia CAD, com a introdução da modelação 3D, especialmente na modelação de sólidos. O CAM seria cada vez mais usado noutras indústrias, como a indústria de moldes e ferramentas, e suportaria tecnologias mais recentes na área do comando numérico, como a utilização de NURBS[4] ou a capacidade de maquinar a alta velocidade[5]. A implementação de algoritmos de deteção e planeamento de desbastes secundários[6] nasceria também nesta década [8]. A nível de comunicação, desde a sua forma original – a fita perfurada – outras foram desenvolvidas. Com a introdução do CNC, nos anos 70, passou a ser possível gravar programas de CN na memória da unidade de computador da máquina. Para a transferência de dados utilizar-se-iam protocolos como o RS-232 (mais conhecido como porta série), ou as disquetes de formato grande. Criou-se também o DNC (*Direct Numerical Control*), que permite a transmissão do programa bloco-a-bloco, ou partes de programas [1] [8]. Mais tarde poder-se-iam aplicar as redes de computadores à comunicação com máquinas CNC.

As marcas de *software* CAM, a par das novas aplicações que o mesmo ganhou nos últimos 20 anos, viriam a ser segmentadas conforme o mercado. É o caso do CircuitCAM (lançado em 1991), do ShipBuilder (cujo início data de 1990), ou da 3Shape, que se aplica ao fabrico de implantes dentários (lançado em 2000). Por outro lado, os produtos mais antigos (Mastercam, CATIA e SurfCAM), oriundos de uma fase em que o CAM estava mais associado à máquina com controlo CNC e não tanto à aplicação industrial, começaram a integrar cada vez operações[7] [6] [9].

---

[3] Não confundir com *feature based machining*, um conceito mais moderno, e que será referido posteriormente.

[4] *Non Uniform Rational Basis Spline*, ou NURBS, é a resposta à necessidade de se representar superfícies livres por equaçõs polinomiais.

[5] Alguns *spindles* começariam a ter velocidades acima das 15000 RPM. Estratégias que tirassem partido desta capacidade seriam designadas de estratégias de alta-velocidade.

[6] Entenda-se que desbastes secundários são os que se aplicam quando ainda é necessário remover algum material antes do acabamento final. Estes desbastes secundários normalmente necessitam de uma ferramenta menor.

[7] Faz-se aqui uma distinção entre a aplicação industrial (i.e. indústria naval, aeronáutica, etc.) e a operação em si (i.e. fresagem, torneamento, perfilagem de chapa, etc.).

Baseado no que foi referido anteriormente, pode-se resumir que, desde a sua origem até à década de 2000, os avanços das ofertas CAM evoluíram essencialmente[8] nos seguintes aspetos:

- Utilização da linguagem APT e pós-processamento;
- Utilização de interface gráfica e modelação derivada do CAD;
- Geração de código CN para operações de contorno e operações ponto-a-ponto;
- Criação automática de percursos, através de seleção de geometrias;
- Deteção de zonas de desbaste secundário;
- Estratégias de maquinagem para alta velocidade;
- Protocolos de comunicação DNC e em rede.

Relativamente aos campos de aplicação do CAM, a sua evolução contempla:

- Indústria aeronáutica;
- Indústria automóvel;
- Indústria naval;
- Indústria têxtil e do calçado;
- Produção de móveis e outros produtos em madeira;
- Produção de PCBs[9];
- Indústria de implantes dentários;

No intuito de se estabelecer uma relação entre o que foi desenvolvido desde o início do CAM, até à atualidade, continuar-se-á este estudo, agora com incidência nas tecnologias mais modernas e as últimas aplicações industriais.

Atualmente, a tecnologia que constitui o CAM é desenvolvida para acompanhar as suas próprias aplicações, como por exemplo o CNC e as máquinas onde é aplicado. Em contraponto, o próprio CNC cresce em complexidade e funcionalidade graças ao CAM, que lhe suporta essas funções [10]. Como exemplos disto têm-se as máquinas com mais de um *spindle*, tornos-fresadora, tornos designados *Swiss-style*[10] e robots. A função que suporta a utilização simultânea de mais de uma ferramenta é designada de *multi-tasking machining* [11], e começou, desde cerca de 2010, a ser implementada. Outros desenvolvimentos a nível da máquina CNC incluem a introdução de sistemas de medir com comunicação com o *software* CAM [7] e sistemas de comunicação orientados para o processo e partilha de dados, que permitem, na própria máquina, visualizar a simulação do processo e ter total acesso ao modelo da peça a fabricar. Outro desenvolvimento que provocou grande inovação foi a criação e popularização das máquinas de 5 eixos. Como resposta, o CAM evoluiu no sentido de fornecer estratégias e mesmo módulos de *software* dedicados a essas máquinas, assim como aos seus

[8] De notar que a lista apresentada representa um resumo do estado tecnológico, e que naturalmente algumas marcas teriam avançado além do que está descrito.

[9] Do inglês *Printed Circuit Board*, PCB é a designação que se dá a uma placa de circuito impresso, presente atualmente em virtualmente todos os produtos de electrónica.

[10] *Swiss-style* é um termo usado para designar máquinas-ferramenta projetadas e construídas para precisão superior. Em termos construtivos, utilizam reforços estruturais e sistemas de guiamento superiores. São também conhecidas por disponibilizar várias ferramentas simultaneamente.

processos e produtos: i.e. peças para turbocompressores, pás de turbinas, e implantes dentários. Esta conjugação permite agora que operações de elevada complexidade sejam programadas [5] [6] [12].

2.3 – Apoio à tomada de decisão e planeamento tecnológico

No respeitante ao suporte de tomada de decisão e planeamento tecnológico, têm havido duas principais tendências. Uma no sentido de a tornar mais expedita e com menos intervenção por parte do utilizador, e outra no sentido de a tornar mais completa e com melhores resultados [13].

No primeiro caso tem-se feito esforços para uma melhor interação com o *software* CAD e, particularmente, com os seus modelos, evitando gastar-se tempo na identificação de *features*[11] [14]. Algumas aplicações, como o Mastercam ou HSMworks, podem integrar-se parcialmente com outros *software* de CAD, funcionando de uma forma modular. Por outro lado, no caso do CATIA, a integração é total, pois nesse caso, tanto CAD e CAM partilham da mesma estrutura de *software* [6] [15]. Ainda acerca da utilização dos modelos, começam a integrar-se algoritmos de deteção de *features, ensinamento* de *features*, em que o utilizador pode tipificar topologias e torná-las passíveis de reconhecimento automaticamente (*feature teaching).* A mesma tecnologia está a ser aplicada à escolha de operações. O *operation teaching* consiste em usar *features* como uma base que define uma estratégia tipo. A parametrização[12] de entidades geométricas, que é hoje em dia dominante nas ofertas de *software* CAD, ainda não está presente, com a mesma consistência, no *software* CAM. Esta funcionalidade permite modificações rápidas e flexíveis à geometria, sem comprometer a configuração dos percursos programados, e sem necessitar de quebras no ritmo de trabalho. *Software* como o PowerMill ou Siemens NX possuem esta funcionalidade. Evoluções na estrutura do *software* e utilização do *hardware* contribuem também para que a utilização do CAM seja mais produtiva [7] [12].

No segundo caso – qualidade do apoio à tomada de decisão e planeamento tecnológico – a concentração de *know-how* levou a que se criassem estratégias cada vez mais adaptadas a tipos diferentes de construções geométricas (i.e., distinguindo por exemplo uma topologia unidirecional de uma radial), assim como algoritmos de otimização do acabamento (como por exemplo o *toolpath refinement* do Mastercam) ou velocidade de desbaste (*optirough*, do mesmo, ou *adaptive area clearance*, da Delcam). Muitas marcas oferecem também combinações destas e outras estratégias, numa funcionalidade designada de *hybrid machining* (ou maquinagem híbrida) [6] [5]. Neste ponto insere-se também o que se pode considerar "a funcionalidade crucial da década de 2000": a simulação do processo de maquinagem [16]. Esta funcionalidade tem tido uma constante evolução. Inicialmente consistia na simulação do movimento da ferramenta, e mais tarde da remoção de material no processo de maquinagem, com posterior

[11] *Features* é um termo usado em CAD que pode ser interpretado como uma construção geométrica parametrizável com uma forma normalmente associada a uma função.

[12] Parametrização consiste em caracterizar a dimensão de uma entidade geométrica de forma associativa.

comparação entre a forma pretendida e a forma obtida [6]. Com a evolução da programação *offline*[13], também a simulação da máquina – através da criação de uma máquina virtual – passou a ser feita com o *software* CAM [7] [12]. Graças a esta tecnologia, passou a ser possível fazer deteção de colisão assim como estimativas mais realistas do tempo de ciclo, e inclusivamente, simular o movimento e dinâmica da ferramenta, avaliando a sua resposta aos esforços de maquinagem. Inicialmente, a simulação das operações na máquina virtual, eram feitas apenas com os percursos computados pelo *software*. No entanto a tendência atual é a de usar o próprio código de CN para conduzir a simulação, extinguindo qualquer risco. Conjugando esta funcionalidade com os novos sistemas de comunicação que, como referido, permitem a partilha de dados, o próprio operador poderá verificar a operação antes de a executar. Ainda no tema da simulação, outra evolução importante é o da utilização de sistemas de validação e verificação do código de CN [12].

2.4 – Aplicações industriais

O crescimento do número de materiais passíveis de ser trabalhados, assim como do tipo de máquinas-ferramenta disponíveis para o fazer, criou novas oportunidades para o CAM: este foi tendo um crescente número de aplicações industriais [1]. Além da sua primeira implementação – a da indústria aeronáutica, nos Estados Unidos da América – em meados de 1960, o CAM foi também aplicado à indústria automóvel e naval [17]. Já na década de 70, o CAM começou a ser usado na indústria têxtil e de calçado [18]. A partir desta época, na década de 1980, o CAM passou a ser utilizado em qualquer indústria que recorresse a máquinas com CN. Exemplo são a indústria das máquinas para trabalhar madeiras, ou a indústria eletrónica [17] [19]. Relativamente às aplicações do CAM, na última década, estas expandiram-se para a indústria automóvel de alta-performance e alta-competição (como exemplo: a Fórmula 1), relojoaria e dispositivos médicos (também como exemplo: próteses e reconstituição óssea). O CAM continua também a marcar a sua presença na indústria dos implantes dentários [10] [20] [21].

2.5 – Estrutura de software

A nível do desenvolvimento da estrutura do *software* em si, os últimos esforços têm sido no sentido de paralelizar[14] os algoritmos de geração de percurso, e também o próprio programa. Isto permite tirar partido dos vários núcleos que, na última década, constam num processador. Este aproveitamento reduz o tempo de cálculo, pós-processamento, e simulação em até cerca de 60% [12]. O *software* CAM da marca HSMworks fornece inclusive uma aplicação que permite aos utilizadores testar os seus

[13] Programação *offline* consiste em programar programas de CN num terminal desligado da máquina.

[14] Paralelização consiste em programar com vista correr o programa em simultâneo em vários núcleos de um processador.

próprios computadores, atestando o impacto de um CPU moderno [15]. O *multi-tasking* é também uma vantagem dessa arquitetura de vários núcleos, pois o utilizador poderá, de uma forma genérica (i.e., para todo e qualquer *software*) executar várias aplicações simultaneamente (por exemplo, enquanto está uma simulação ou pós-processamento a decorrer no *software* CAM, o utilizador pode estar a trabalhar noutro modelo num outro *software*, por exemplo, de CAD). Por outro lado e de uma forma mais específica, por exemplo com o *software* PowerMill, da Delcam, a paralelização do programa permite continuar a utilizá-lo para trabalhar numa parte do modelo, enquanto outra parte está submetida a cálculo de percursos. Já na área da interface com o utilizador, a par da performance gráfica que tem sido constantemente otimizada para ter requisitos de *hardware* mais moderados, a tendência é a de simplificar a interação com a interface gráfica (GUI - *graphics user interface*), proporcionando proximidade imediata entre os comandos mais usados e o utilizador básico, mantendo, ao mesmo tempo, acesso aos comandos avançados. Há também esforços no sentido de a tornar mais intuitiva, e também ajudar o utilizador a interagir com a representação 3D do modelo. A customização dos menus e barras de comandos é também algo que tem vindo a ser implementado [5] [6] [7].

2.6 – Mercado CAM

Relativamente ao mercado de *software* CAM, em linha com o que já foi referido, constata-se que a utilização de sistemas CAM tem ganho cada vez maior relevância no mundo industrial. Não só pelas vantagens imediatas em adotar, na fase de planeamento de processos, métodos de trabalho mais eficientes, mas também pela atual possibilidade de tirar proveito dessas técnicas em novos mercados, mencionados anteriormente: implantes dentários, relojoaria, entre outros.

Face a este crescimento, os fornecedores de *software* de CAM têm reagido positivamente, estabelecendo-se como influenciadores do mercado de equipamento CNC, assim como de CAD, e, consequentemente, do mercado de PLM (*Product Lifecycle Management*). Havendo também interesse e necessidade, por parte dos criadores do *software*, em acompanhar a evolução tecnológica da maquinagem em si, esta relação tornou-se cíclica e interdependente. As marcas de maior sucesso optam assim por atualizar constantemente os seus produtos [10].

O mercado de *software* CAM é também propício ao fenómeno recorrente da fusão empresarial. Marcas mais prevalentes assimilam desta forma o *know-how* emergente, muitas vezes levando à integração de várias funcionalidades, i.e. CAD, CAM ou CAD/CAM. É o caso da HSMworks[15], adquirida pela Autodesk em 2012; da Tecnomatix[16], adquirida pela Siemens, ou da SylvieXpert[17], adquirida pela CNC *Software* – e agregada no produto Mastercam com a designação *Swiss Expert* [6] [16].

[15] A empresa HSMworks fornece soluções CAM bem cotadas no mercado. Foi este o motivo da sua aquisição, pois a Autodesk até a essa data não possuía nenhuma oferta CAM.

[16] A Tecnomatix especializou-se em ambientes e programas de simulação de operações CAM.

[17] A SylvieXpert é uma marca de *software* CAM dedicada à utilização de máquinas *Swiss-style*.

A exclusividade de determinadas funções é uma componente facilmente observável no mercado CAM, fruto não só da competitividade que o próprio gera, mas também da já mencionada especialização. Muitas marcas assumem inclusive que não lhes é possível fornecer ofertas para todas as aplicações que o mercado procura. Este facto, aliado à crescente facilidade em desenvolver um produto digital[18], contribuiu para o crescimento das várias empresas de *software* CAM, assim como a criação de novas empresas [12] [22]. Na Figura 2 é apresentado o resultado de um estudo de mercado dos principais fornecedores de *software* de CAM em 2010.

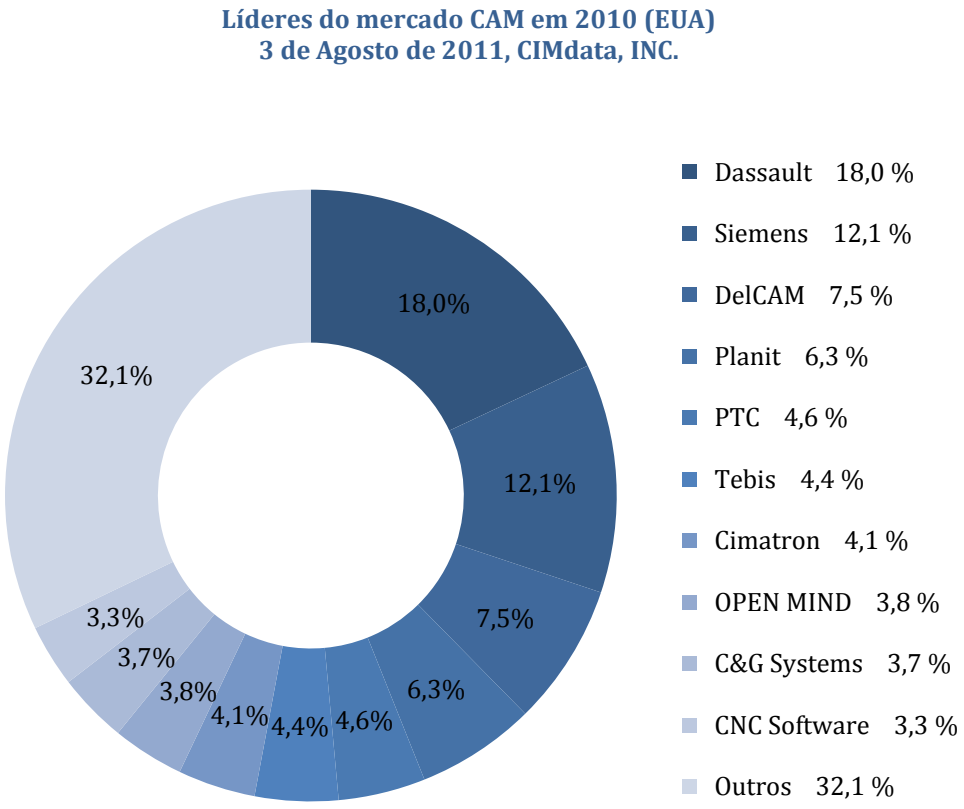


Figura 2 - Líderes do mercado CAM em 2010 - vendas diretas de licenças e prestação de serviços [22].

O domínio de *software* CAM exige o desenvolvimento paralelo de um mercado já criado em outros segmentos do *software* PLM: o da formação profissional. Denunciando a previsível dificuldade de adaptar a componente humana a novas ferramentas de trabalho, as empresas esforçam-se cada vez mais no sentido de atingir a máxima eficiência dos seus investimentos em novas tecnologias. Seja por falta de quadros técnicos devidamente especializados, ou pelo investimento associado à formação, esta está presente cada vez mais, e de forma mais competitiva. Este crescimento tem um efeito visível na qualidade do corpo técnico de uma empresa, e nas repercussões salariais da mesma [3] [17]. As marcas de *software* CAM, acompanhando

[18] Hoje em dia, com a vulgarização do computador e facilidade em obter aplicações para programação, assim como literatura de apoio, é cada vez mais fácil desenvolver um produto digital.

esta procura de formação, começam a incluir pacotes de treino e aprendizagem. É por exemplo o caso da Delcam e da Siemens [5] [7] [10].

O mercado CAM continua a ser a base para um aumento de competitividade no plano internacional. De acordo com Stan Przybylinski, diretor de pesquisas da CIMdata: " [nos EUA] 2011 foi um ano de recuperação no sector fabril, como se observa nos níveis de encomendas das ferramentas CNC e investimento nos *software* associados." Já de acordo com Alan Christman, presidente da CIMdata e autor da *NC Market Analysis Report*, "2011 foi um ano excelente para fabricantes e marcas de *software* de CN. Muitas empresas mostraram crescimento saudável. O contínuo esforço e crescente importância de potências produtoras como a China e outras economias emergentes resultará no aumento dos investimentos em tecnologias como o CAD, CAM, e outros segmentos do mercado PLM. (…) Registamos também esforços para que se reinvista na indústria dos Estados Unidos (em detrimento da asiática), e que se traga a produção fabril de volta [para os EUA], o que necessitará de maiores investimentos nestas mesmas tecnologias, para que possamos competir com economias de menor quadro salarial."(Figura 3) [23].

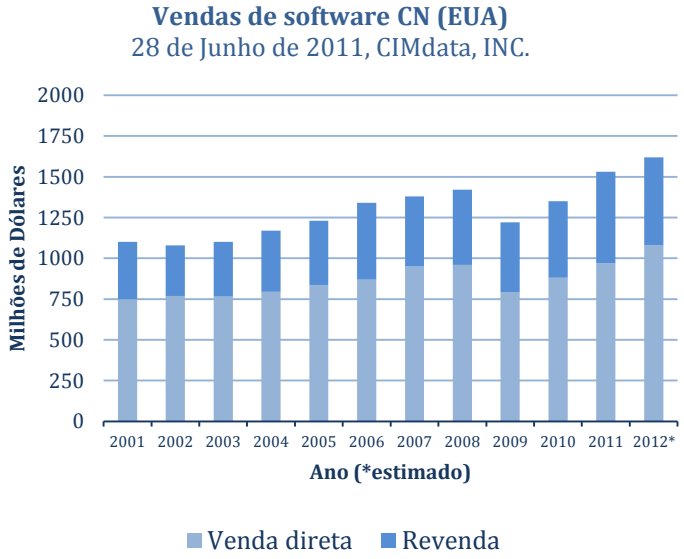


Figura 3 – Crescimento das vendas de *software* CN (EUA) em 2011 [23].

A recolha de informação feita e exibida neste capítulo, embora demonstre o estado-da-arte de uma forma abrangente, não o faz com detalhe. Como tal, para melhor comparação e entendimento do processo evolutivo do estado-da-arte, e, baseado nas referências apresentadas e na compreensão geral da sua leitura e pesquisa, apresenta-se, na Figura 4, um diagrama em que se representam diversos conceitos tecnológicos e funcionais.

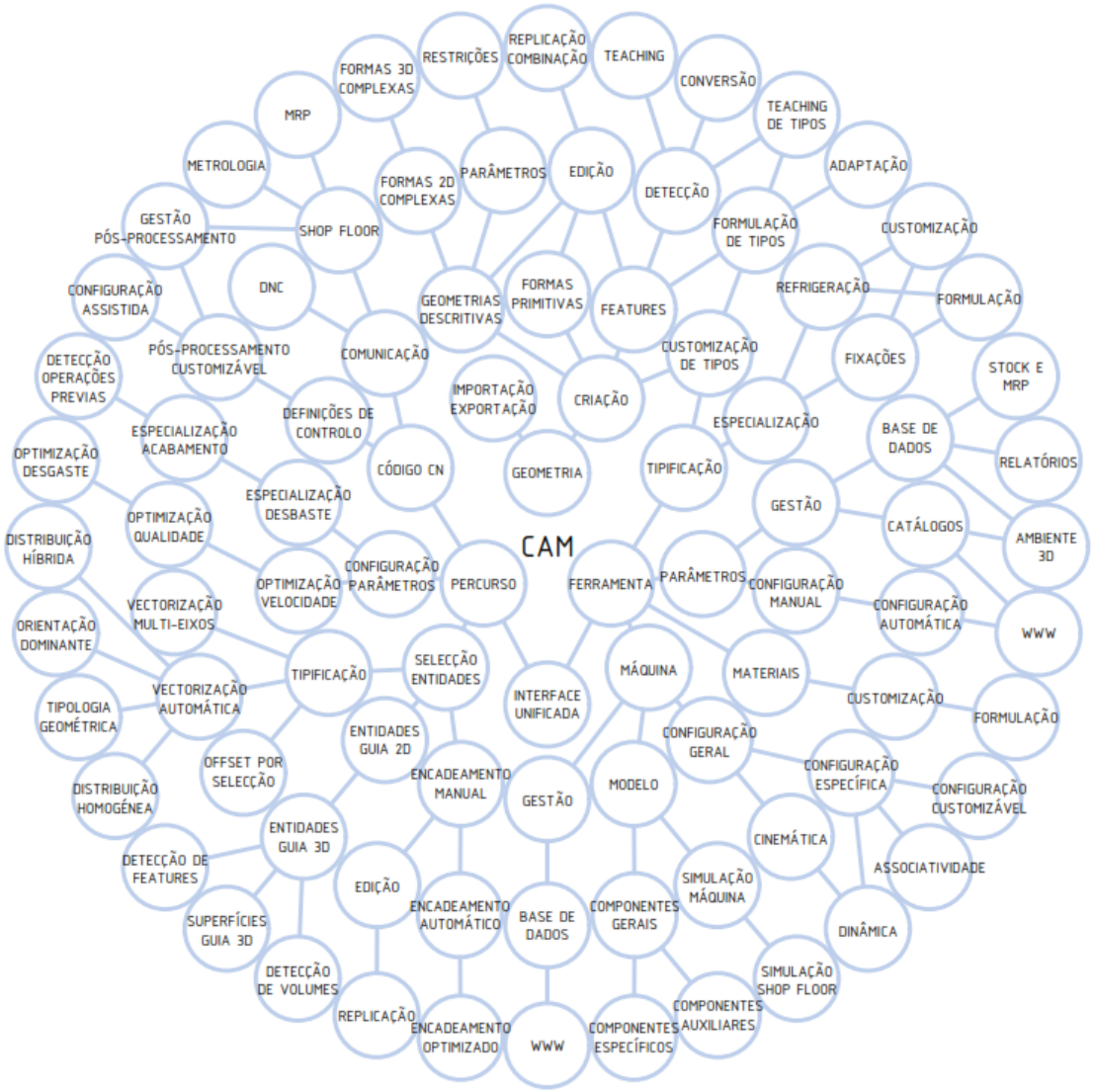


Figura 4 - Mapa tecnológico geral: estado-da-arte.

Esta representação gráfica de um conjunto de posições tecnológicas permite, entre diferentes ofertas de *software* CAD/CAM, uma comparação fácil das suas capacidades, e um entendimento claro da vocação de cada um. A escolha deste tipo de *software* usando esta via de comparação pressupõe o conhecimento dos requisitos funcionais – do local onde o mesmo será utilizado – assim como a capacidade de traduzir demonstrações da própria marca e/ou fichas de informações dos programas. Isto é, saber reconhecer funcionalidades iguais com nomes distintos.

Estes conceitos dividem-se por níveis de complexidade, com uniões entre si, procurando relacionar as influências que determinadas tecnologias têm nas evoluções seguintes, e/ou sugerir uma ordem evolutiva lógica.

Nos níveis interiores encontram-se, tipicamente, tecnologias que sustentam a própria existência de CAM, como, por exemplo, a geração de **código de comando numérico** a partir da **criação de percursos** com recurso a **geometrias** e **definições de ferramentas**.

Nos níveis intermédios dispõe-se evoluções de tecnologias mais simples (as anteriores) – como no caso das **formas 2D complexas** ou **configuração automática** de parâmetros de ferramentas – ou implementação de tecnologias de valor acrescentado, como a **gestão de bases de dados** de ferramentas, ou a **simulação da máquina virtual**.

Nos níveis exteriores encontram-se os conceitos associados ao mais avançado do estado-da-arte, ou as últimas evoluções de tecnologias que têm vindo a ser implementadas desde o início do CAM.

Para resumir este subsecção, o estado-da-arte do CAM pode ser descrito nos seguintes pontos:

- Existência de um motor de CAD completo e robusto, que transforme o produto CAM em um CAD/CAM, ou possibilidade de articular ambos[19] de forma parcial ou total;
- Possibilidade de ensinar a criação de operações e a gestão de ferramentas, de modo a adaptar novos tipos a determinadas *features*;
- Gestão de ferramentas altamente suportada por ambiente 3D, e altamente customizável;
- Customização total da máquina virtual, modelação completa em ambiente 3D com simulação do *shop floor*[20], e utilização máxima das variáveis cinemáticas e dinâmicas da mesma;
- Panóplia de operações, com deteção de orientação geométrica, alta sensibilidade topológica, com deteção de volumes (essencialmente prismáticos), e deteção robusta de *features*, e articulação de mecanismos de ensinamento programados em ambiente CAD;
- Diversas otimizações de velocidades e perfis de corte, que permitem, por exemplo, valorizar a velocidade ou o acabamento final, inclusive apontando para rugosidades planeadas;
- Gestão de informação altamente dinamizada com recurso a redes de comunicação profissionais – do inglês, *collaboration* – assim como bases de dados de máquinas, ferramentas, fixações, materiais, e inclusive, processos de fabrico;
- Interfaces altamente desenvolvidas e facilitadoras da experiência de utilização do *software*, com vista a maximizar o conceito de *workflow*.

[19] De facto, no caso do Mastercam X5, é possível articular todas as funções CAM com o *software* Solidworks, utilizando todas funcionalidades CAM a partir desse *software* CAD.

[20] *Shop floor* é um termo que distingue o local onde se trabalha normalmente com um computador, do onde são maquinadas as peças, ou seja, onde estão os centros de maquinagem. Neste caso, trata-se de um *shop floor* simulado, isto é, ambiente virtual.

Por fim, na Figura 5 representa-se o mapa já apresentado na Figura 4, mas agora com destaque para as funcionalidades incluídas no *software* Mastercam X5.

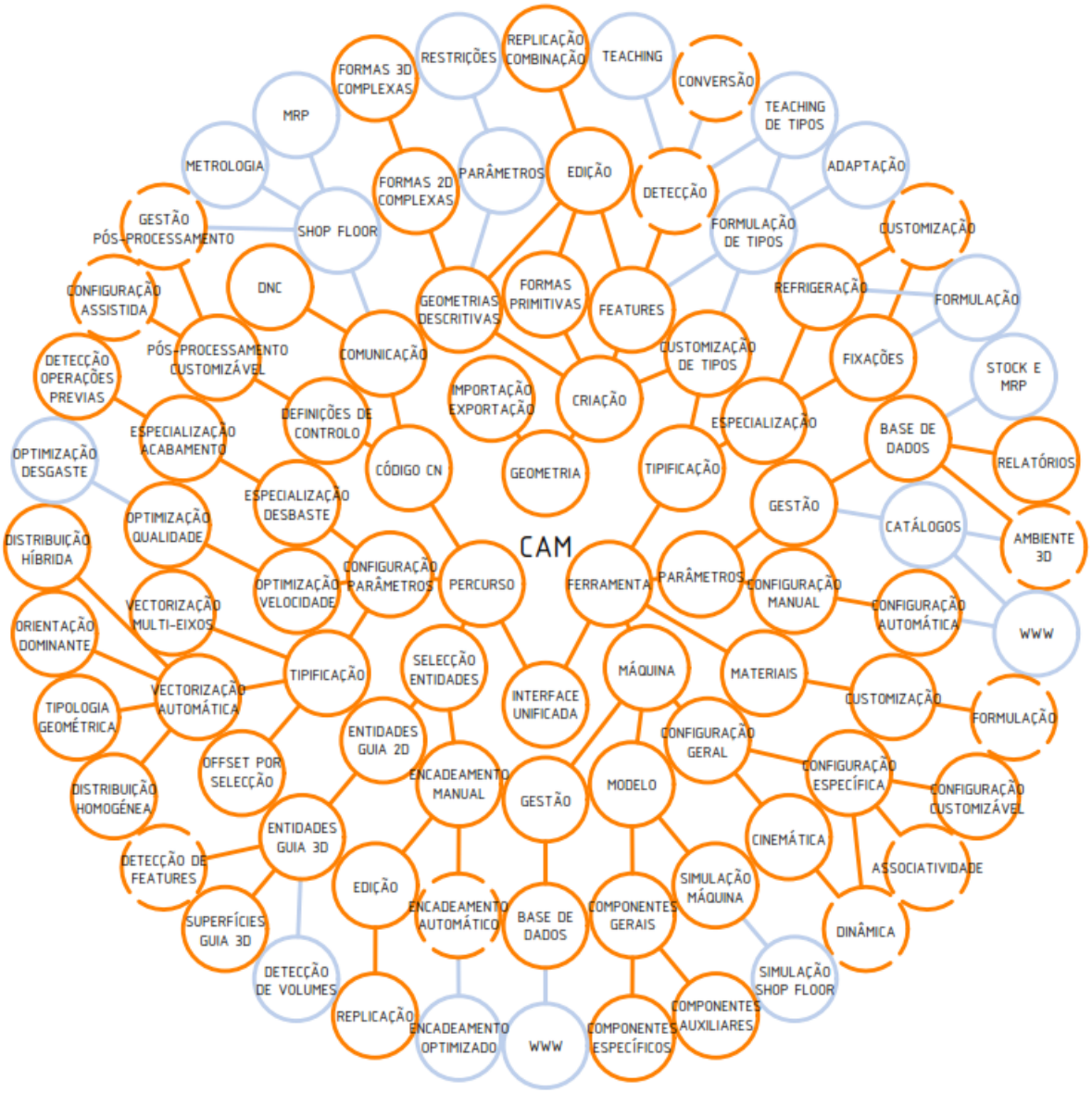


Figura 5 - Mapa tecnológico: Mastercam X5.

Observando a Figura 5 percebe-se a maior vocação do Mastercam X5 para a criação de operações, e não tanto para a criação de geometria (afastando, de certa forma, o conceito de programa de CAD/CAM). De facto, a fraca implementação de algumas funcionalidades (representado a traço interrompido), associadas à criação de *features* assim como a falta de parametrização de geometrias descritivas e restrição geométrica[21], exigirão que se disponha de uma solução CAD devidamente equipada.

[21] Por restrição geométrica entende-se como a criação de condições que influenciam os graus de liberdade de determinadas entidades, e, como tal, o seu posicionamento. A título de exemplo: condições de paralelismo, tangência, perpendicularidade, concentricidade, entre outras.

3 – *Software* de CAM: Mastercam X5

À data da realização deste trabalho o software de CAM, Mastercam X5, é o antepenúltimo lançamento da série X, e é produzido pela empresa CNC *Software*, INC. Fundada em 1983, dedicou-se de raiz ao mercado do *software* CAM. A série X, iniciada no ano 2005, representa a passagem do *software* para ambiente Windows (as versões anteriores eram desenvolvidas para o sistema DOS), sendo caracterizada por:

- Uma interface de mais fácil utilização;
- Possibilidade de customização de menus e barras de ferramentas;
- Melhor utilização de ficheiros STL;
- Melhor suporte geométrico e ferramental.

A CNC *Software*, INC, desde o lançamento do Mastercam X5, em 2010, tem continuado a evoluir a sua gama de módulos, reforçando o conceito de *software* CAM como um produto configurável.

Como se ilustra no diagrama (Figura 6), o Mastercam X5 contempla a escolha de várias ferramentas de *design* (CAD), aplicações para fresagem, torneamento, roteamento, e corte por fio. A par destas, existem ainda ofertas mais especializadas.

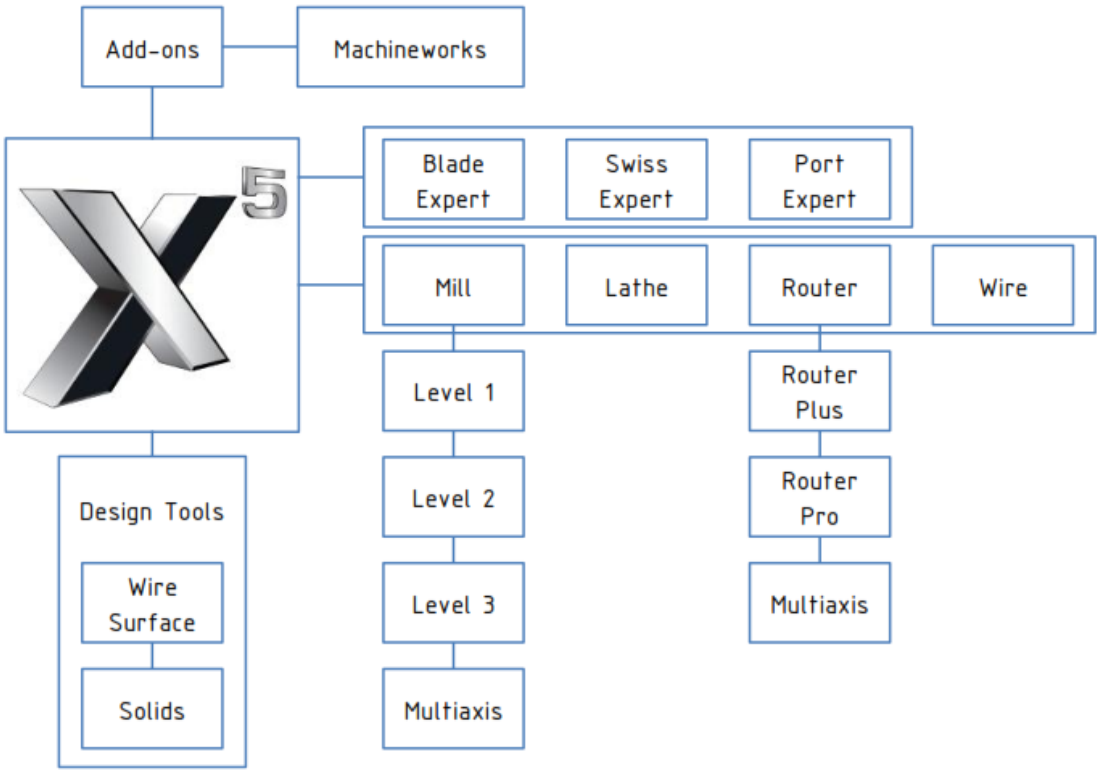


Figura 6 - Mastercam X5 e os seus módulos.

3.1 – Interface

A definição de curvas de aprendizagem moderadas é um requisito na composição de uma boa estratégia de marketing. A facilidade com que é possível aprender e utilizar um produto digital é, neste mercado, um trunfo: uma interface devidamente estruturada é sempre uma mais-valia para o utilizador.

O programa Mastercam X5 tem-na dividida da seguinte forma:

- A área de trabalho, onde se visualiza o modelo e as suas operações;
- A árvore de maquinagem, onde se gerem as operações a realizar;
- As barras de ferramentas, que albergam todos os comandos.

Esta disposição de elementos é flexível. No entanto, não é usada a interface por *tabs* (*tabs* é a designação de uma forma de organizar a interface popularizada pela Microsoft com o *software* Microsoft Office, e que outras marcas de *software* CAD/CAM já implementaram nos seus produtos). A ausência deste sistema não contribui para um *workflow* simplificado. Consequentemente, o utilizador terá de recorrer ao menu frequentemente, o que se resume numa utilização mais morosa do Mastercam X5. Na Figura 7, são visíveis os elementos básicos da interface. À direita, a título de exemplo, mostram-se as barras de ferramentas que estão ativadas. Esta interface poderá levar a que, escolhido o Mastercam X5 como *software* de CAD/CAM, se deva ter em conta a aquisição de um ecrã de maiores dimensões.

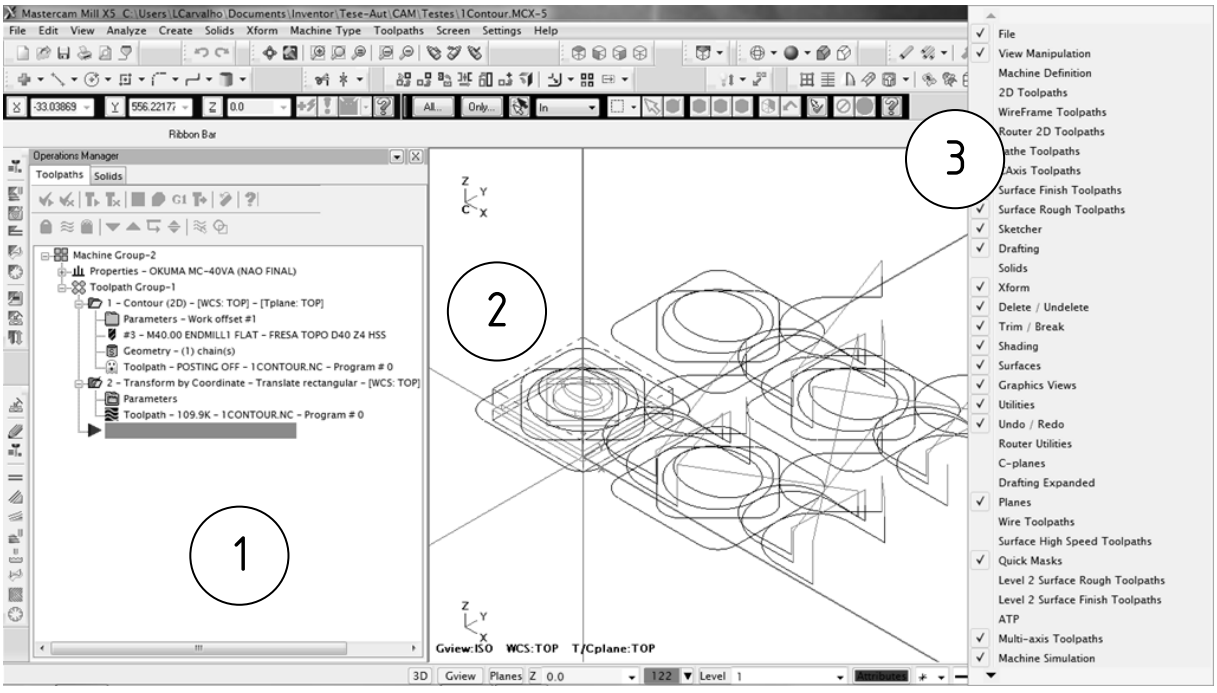


Figura 7 - Da esquerda para a direita: árvore de maquinagem (1); área de trabalho (2); menus (3).

3.2 – Funcionalidades

O Mastercam X5 contempla um conjunto de funcionalidades que abrange tanto aspetos mais básicos do *software* CAD/CAM, como outros mais avançados.

As funcionalidades presentes nesta versão dividem-se nos seguintes grupos:

A. Gestão de entidades geométricas

B. Funcionalidades CAM

C. Gestão de máquinas virtuais

D. Gestão de ferramentas virtuais

E. Calculador de estratégias de fresagem[22]

F. Gestão de operações

G. Estruturação de dados

Estas funcionalidades serão, neste subsecção, explanadas, e no final, resumidas e contrapostas com o que se entende por um sistema CAD/CAM ideal. Esta comparação, terá em consideração as capacidades atuais dos sistemas CAD/CAM, e, em virtude da possibilidade de unir sistemas dedicados a CAD com sistemas CAM[23], considerará igualmente este potencial.

As funcionalidades acima listadas não incluem a gestão da interface do *software*, que, por se considerar básica e parte do *standard* do sistema operativo Microsoft Windows, não requer especial atenção.

A. Gestão de entidades geométricas

O Mastercam X5 permite criar, importar, e editar geometrias simples e complexas. O motor geométrico, baseado no sistema *Parasolid*[24], tem a capacidade de modelar tanto sólidos como superfícies. A partir destas, o motor permite criar volumes (e

[22] No âmbito deste estudo, e em virtude do trabalho com um centro de fresagem OKUMA, esta descrição concentrar-se-á nas funções CAM para fresagem.

[23] De facto, tendo como exemplo o *software* Solidworks, que permite trabalhar diretamente com o Mastercam ou outras marcas (HSMworks, p.e.), este fator terá de ser tido em conta.

[24] *Parasolid* é a designação de um motor geométrico avançado que suporta criação de sólidos por BREP ou CSG, e funcionamento por *features.*

consequentemente, sólidos), e vice-versa. Pode ainda realizar operações booleanas[25]. Todas as geometrias criadas são passíveis de ser replicadas usando sistemas cartesianos ou polares. A exportação de geometrias está limitada aos formatos neutros mais populares (STEP, IGES e STL, etc), e outros, proprietários, mas também comuns.

As geometrias são interpretadas pelo programa como entidades. São, em cada modelo, seriadas, e, à discrição do utilizador, divididas por níveis[26].

A modelação por *features* não é tão completa quanto a presente noutras ofertas CAD e inclusive CAD/CAM. No entanto, para a maioria das operações de modelação, está presente (p.e., não existe a *feature* de furo, mas é possível fazê-lo por corte-extrusão[27], com ajuda de geometrias descritivas).

A parametrização das entidades geométricas é mínima. O Mastercam X5 não utiliza o conceito de *constraint* – ou regra – que permite estabelecer relações de paralelismo, união ou tangência, entre outras. As características das entidades geométricas não são geridas globalmente, mas sim localmente, o que se torna pouco prático (p.e., se uma curva for criada entre um ponto A e um ponto B, e se a posição de B for alterada, a curva não "seguirá" esse ponto).

Existe ainda possibilidade de detetar *features*, mas esta funcionalidade está limitada à deteção de cilindros e *fillets*. A sua utilidade é reduzida por esta limitação.

B. Funcionalidades CAM

A versão do Mastercam usada neste estudo contém módulos de *software* para quatro tipos de trabalho de remoção de material: fresagem, torneamento, roteamento e corte por fio.

Havendo, para cada um destes tipos de trabalho, distinções a nível da maquinaria e ferramental utilizados[28], assim como nas operações a realizar, também as opções do programa se alteram. Desta forma a interface altera-se e adequa-se a cada uma das operações, ocultando comandos que não terão efeito em virtude de outros mais pertinentes.

C. Gestão de máquinas virtuais

As máquinas virtuais desempenham a função de representar os aspetos físicos mais imediatos das máquinas reais, e as capacidades dos seus controladores. A máquina virtual é composta por parâmetros e opções, e – ainda que não obrigatoriamente – um modelo CAD da mesma.

Esta funcionalidade, como a sua designação indica, permite criar, editar e importar máquinas virtuais, assim como, entrando também no domínio da funcionalidade G

[25] Operações booleanas consistem na adição, subtração e interceção de volumes e áreas.

[26] Níveis, ou, usando um termo usual de outros *softwares* – *layers* – é uma forma de organizar objetos, que possibilita trabalhar seletivamente.

[27] Corte-extrusão é uma *feature* em que uma extrusão não adiciona material ao modelo, mas sim subtrai-a.

[28] As diferenças existentes entre a configuração do programa para operações de fresagem e de roteamento são muito reduzidas, já que, na sua essência, existem entre ambas várias semelhanças.

(estruturação de dados), articular todas as informações com as bases de dados das outras partes do *software* (i.e., ferramentas, ficheiros de controlo, pós-processadores, etc.).

A gestão de máquinas virtuais é crucial na medida em que é responsável pela tradução dos caminhos gerados nas operações que o utilizador define. Logo, especial atenção lhe deve ser dedicada, mais especificamente nas definições de controlo e pós-processamento (ver capítulo 5).

No Mastercam X5, esta funcionalidade é a base para a criação de operações, e as mesmas não podem existir sem antes estar escolhida uma máquina virtual. Por vezes, este modo de funcionamento, torna a sua utilização pouco expedita.

D. Gestão de ferramentas virtuais

O Mastercam X5 inclui um simples mas eficaz gestor de ferramentas, que, em semelhança com as funcionalidades anteriores, permite também criar, editar e importar ferramentas virtuais. À semelhança das máquinas virtuais, uma ferramenta virtual é apenas a representação de uma suposta ferramenta. A gestão é meramente a nível da ferramenta singular, já que o *software* não possui gestão de quantidades (i.e., não gere *stock* ferramental).

A gestão de ferramentas contém também a definição de formatos standard, formatos especiais e customizáveis, que se adaptam e são, conforme definições das operações em causa, automaticamente escolhidos.

No respeitante a bases de dados, esta gestão permite agrupar várias definições de ferramentas e gravar num ficheiro uma determinada escolha (p.e., que represente a disponibilidade em uma oficina), e inclusive lançar relatórios dessas definições.

E. Motor de cálculo de estratégias de fresagem

O motor de cálculo do Mastercam X5, especificamente para estratégias de fresagem, contempla o cálculo de percursos para formas simples (2D), formas mais complexas (3D), ou que inclusive necessitem de maquinagem com 5 eixos (ou seja, utilizando máquinas-ferramentas com 5 eixos de movimento).

As capacidades deste motor, resumidamente, dividem-se em cálculo de percursos para desbaste, cálculo de percursos para acabamento, e cálculo de percursos transformados e/ou replicados. O Mastercam X5, nesta funcionalidade, divide-os ainda conforme aproximação às formas a obter por maquinagem[29].

[29] Por formas a obter por maquinagem entende-se como o que se pretende obter no final.

O motor de cálculo do Mastercam X5 apresenta uma série de opções que permitem adaptar o percurso às suas preferências, como por exemplo, a qualidade que se pretende obter, ou que tipo de entrada e saída terá o percurso.

O calculador de percursos para acabamentos pode funcionar com deteção de desbastes prévios. Esta funcionalidade permite encadear ambas as operações (desbaste e acabamento), a partir das definições de precedência[30]. Consequentemente, o percurso de maquinagem de acabamento será melhor calculado e alterado para uma menor remoção de material.

F. Gestão de operações:

A gestão de operações, no Mastercam X5, consiste essencialmente na sua organização, padronização, e definição de valores base. Enquanto que o calculador de percursos é a base da criação de operações, esta funcionalidade trata dos parâmetros base com os quais esta assenta. Ou seja, com o gestor de operações, o utilizador pode restringir e/ou predefinir alguns parâmetros que serão fornecidos ao motor de cálculo.

A gestão de operações inclui também a importação e exportação de operações, para que, por exemplo, se criem operações-tipo[31].

Por fim, esta funcionalidade engloba a simulação de operações, com cálculo de tempos de ciclo, e deteção de colisões. Embora o Mastercam X5 não calcule percursos em virtude da não-colisão, existe grande utilidade na existência deste detetor. Aqui, a existência de uma definição da máquina virtual que inclua um modelo 3D, é um requisito[32].

G. Estruturação de dados:

Os dados presentes num plano de maquinagem, e que fazem parte da utilização do Mastercam X5, são estruturados em bases de dados.

Esta funcionalidade consiste simplesmente no agrupamento das várias instâncias das informações das máquinas virtuais, ferramentas virtuais, materiais, etc.

Todas as anteriores funcionalidades se articulam com esta última.

[30] Precedência, no Mastercam X5, explica-se como a(s) operação(ões) que definem uma forma em bruto, antes de se dar o acabamento.

[31] Funcionalmente, é uma extensão da predefinição de parâmetros.

[32] A simulação do movimento da máquina e deteção de colisões, recorre a uma aplicação externa incluída com o Mastercam X5, mas que, por não estar totalmente integrada, requer por si só um modelo definido externamente. Para os devidos efeitos, o mesmo modelo CAD poderá ser usado, ainda que necessite de tratamento especial.

3.3 – Estrutura do funcionamento do Mastercam X5

O princípio de funcionamento do Mastercam X5 tem por base a configuração de etapas que sucedem sequencialmente:

I. Definição de geometria
II. Definição da máquina
III. Definição do material a maquinar
IV. Definição das ferramentas
V. Construção e simulação da sequência de operações e obtenção do código final

Só após a definição destas etapas é que um modelo está apto a produzir um código de comando numérico. Designa-se de modelo o ficheiro, de formato *.mcx-5, onde as etapas são desenvolvidas. O modelo constitui um suporte regido pelas opções do programa e definições iniciais da gestão de máquinas virtuais, materiais, ferramentas e operações. O modelo pode existir apenas com informação geométrica, passando a ser visto pelo programa apenas como um modelo CAD e não CAD/CAM.

Com o propósito de explicar melhor o funcionamento do Mastercam X5 considerem-se as etapas I a V como componentes do modelo CAD/CAM.

Na Figura 8 apresenta-se um esquema simplificado do modelo *.mcx-5 (que se designará doravante de modelo ativo). A inicialização do programa abre por defeito o template de um modelo vazio, isto é, sem informação. A informação contida no modelo, e que define um componente, pode ser obtida por duas vias:

- Através do editor interno do programa, acessível através do modelo ativo
- Importada a partir das bases de dados ou outros ficheiros mcx-5 e/ou CAD/CAM

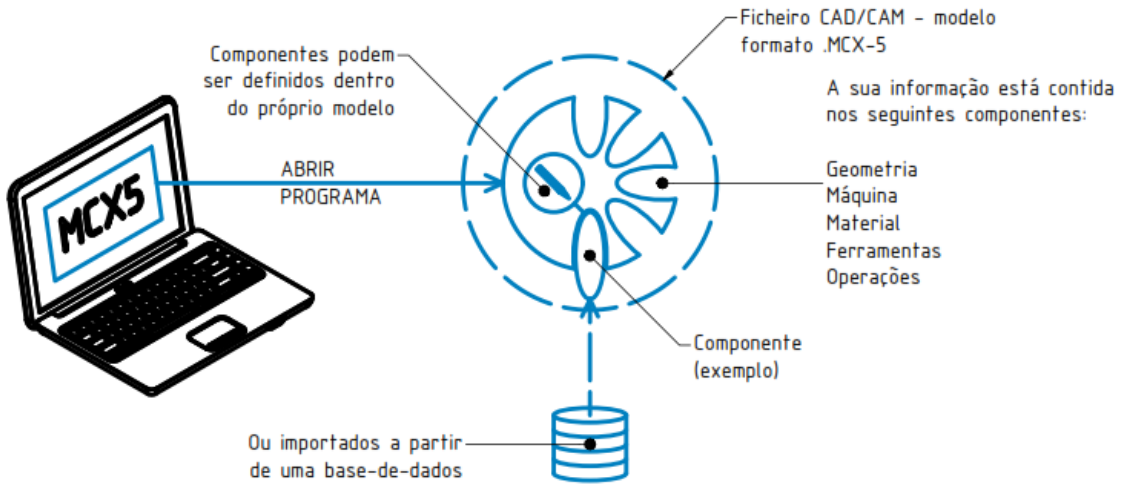


Figura 8 - Representação esquemática do modelo.

A título de exemplo ilustra-se na Figura 8 um componente já definido, assim como a representação de informações do modelo ainda não definidas. Note-se a representação de modelo como um só, a traço interrompido, distinguindo atributos gerais e informações do modelo global, das informações e atributos de um componente em particular.

A interação modelo-Mastercam X5 dá-se a nível de interface, mediante intervenção do utilizador, e a nível interno, através de processos de computação.

Na Figura 9 mostra-se o cerne da interoperabilidade entre componentes. O componente 1, já definido pelo utilizador, fornecerá informações ao longo da definição do componente 2. Estas informações podem ser lidas pelo utilizador e por si manipuladas (p.e., na construção de uma ferramenta especial e utilização de geometrias definidas no modelo ativo), e/ou processadas pelo computador (p.e., na geração de percursos de maquinagem).

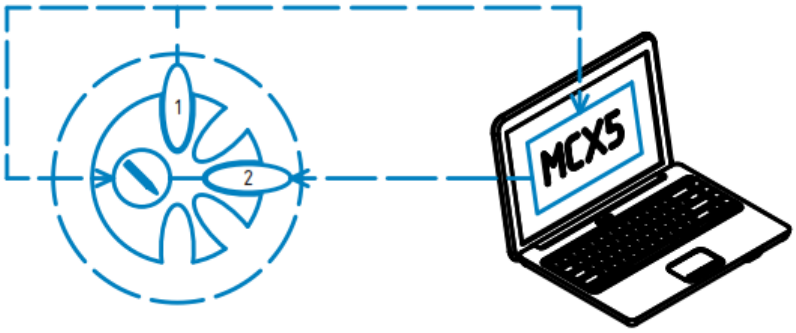


Figura 9 - Recursividade computador-modelo.

A partilha de dados entre modelos é ilustrada na

Figura 10.

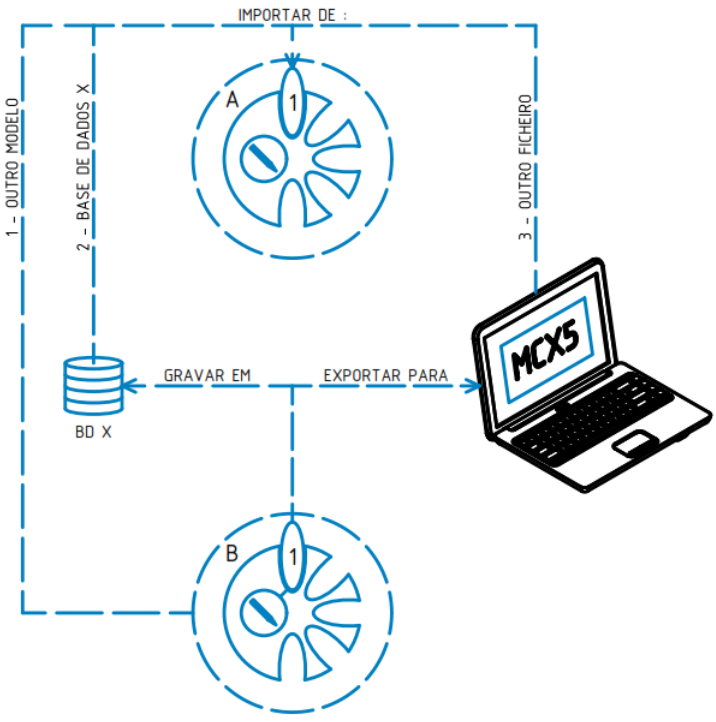


Figura 10 - Funcionamento entre modelos e bases de dados.

A informação do componente 1 do modelo B pode, por motivos de conveniência, ser gravada na base de dados X, ou exportada noutro formato para o sistema operativo do computador. Paralelamente, o componente 1 do modelo A pode importar informação do modelo B:

- Diretamente, por extração de dados do modelo global mcx-5
- Através da instância gravada na base de dados X
- Importando informação de outros ficheiros presentes no computador.

Informações obtidas de bases-de-dados ficam gravadas no modelo como cópias, e a sua edição local diz respeito apenas à cópia, e não ao original.

A sequência de construção dos componentes, representa-se na Figura 11.

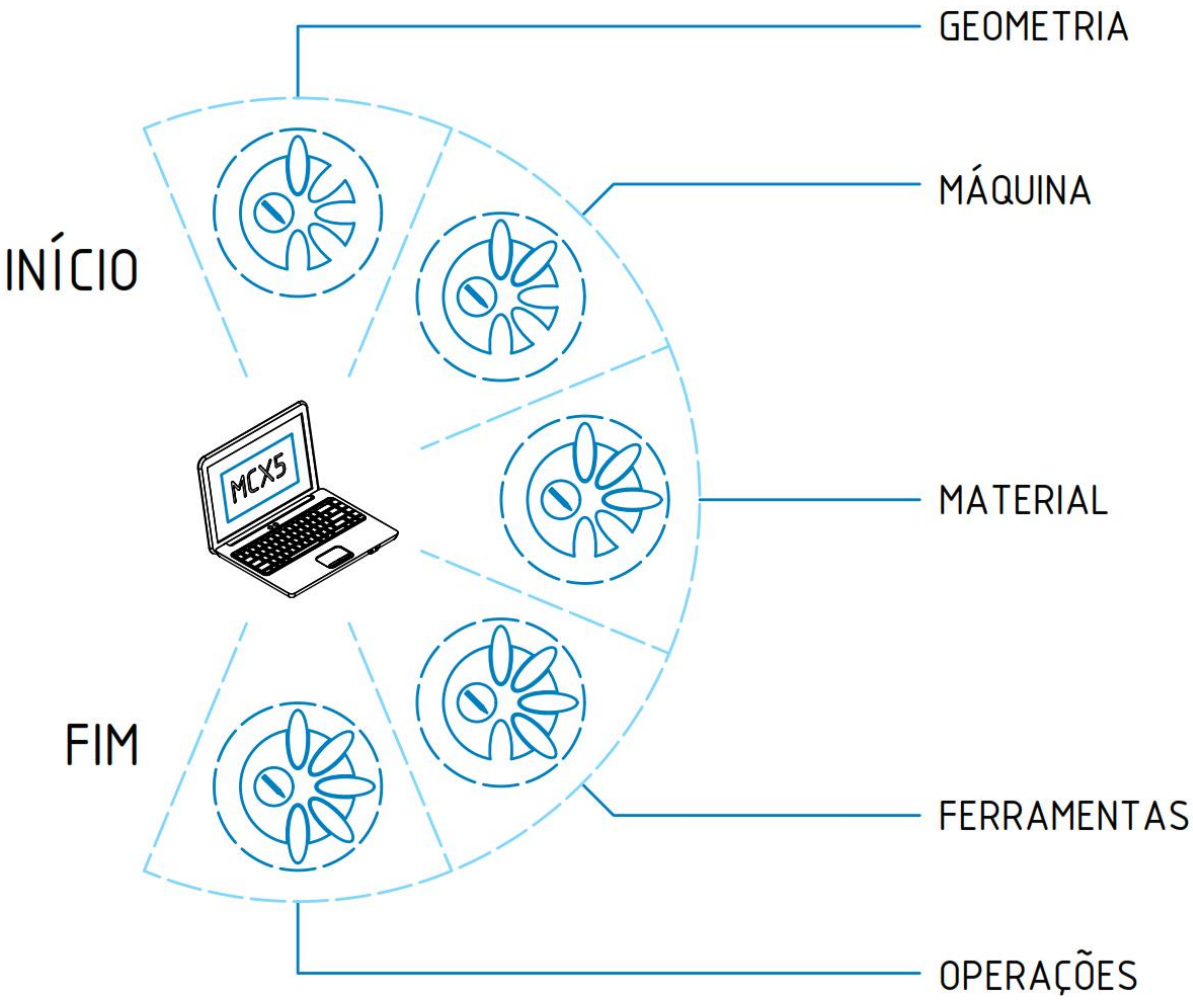


Figura 11 - Sequência de funcionamento dos componentes do Mastercam X5.

Aqui evidenciam-se as 5 etapas essenciais da produção de código de comando numérico no Mastercam X5. Sem a devida informação destes componentes, não é possível fazer uma utilização eficiente do mesmo.

A definição geométrica não requer qualquer definição anterior. É a base do modelo, e como já mencionado, o modelo pode existir funcionalmente, só com esta informação, comportando-se como um ficheiro de CAD.

A estrutura do processo de definição geométrica apresenta-se na Figura 12.

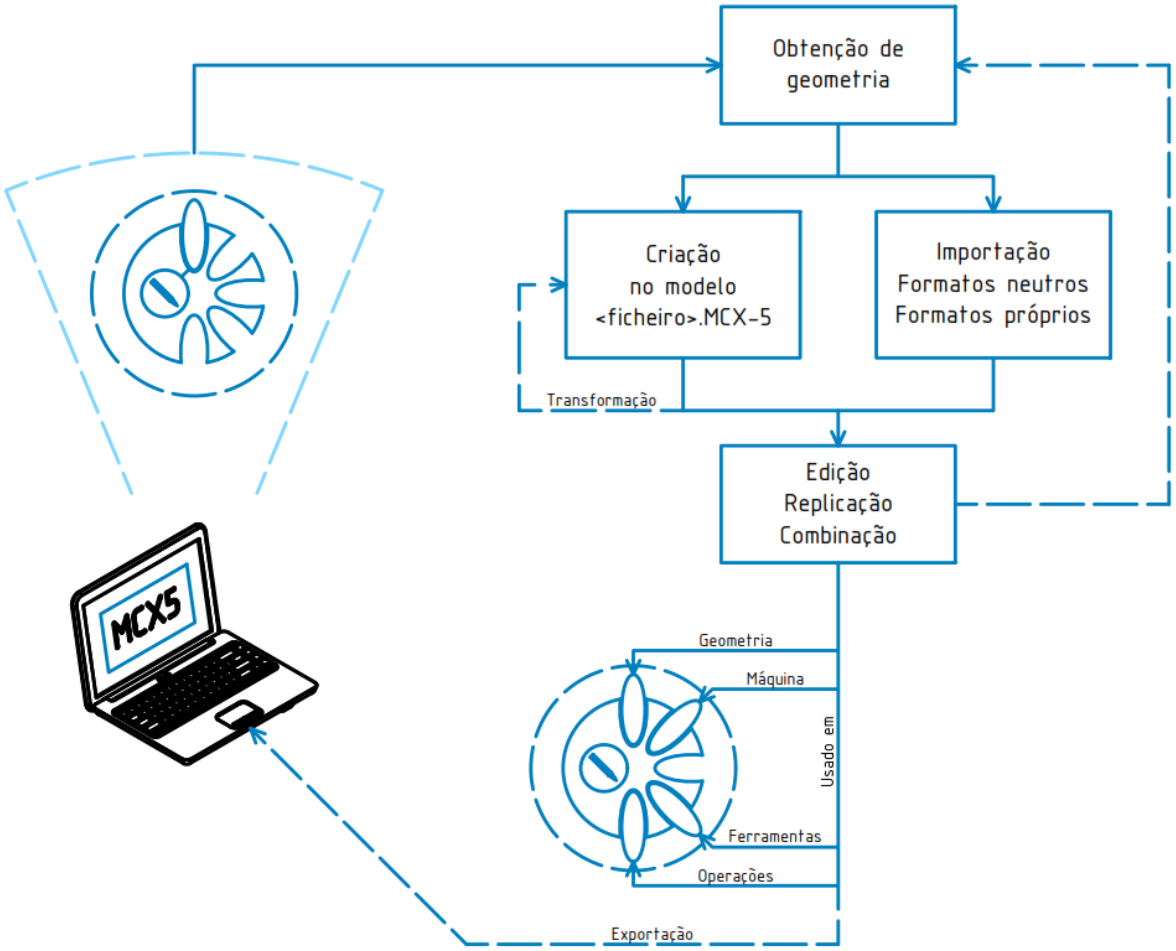


Figura 12 - Componente GEOMETRIA; diagrama de obtenção de formas geométricas.

No Mastercam a gestão de geometrias envolve quatro atividades básicas:

- Criação: o ato de se adicionar alguma forma (recursivo):
  - Através do modelador do programa ou por importação de formatos neutros ou proprietários (inclusive *.mcx-5)
  - Por derivação superfície-sólido ou sólido-superfície
- Edição: alterar ou eliminar partes ou o todo dessa forma
- Replicação: repetir, arbitrariamente, essa forma
- Combinação: obter formas por adição, interseção ou subtração de outras

As informações geradas neste primeiro componente são utilizadas nos componentes acima destacados. A geometria construída no Mastercam X5 é exportável também em formatos neutros e formatos proprietários.

Após a definição das geometrias a maquinar, o utilizador deve iniciar o processo de configuração da máquina virtual, presente no componente máquina.

Do ponto de vista de funcionamento de *software*, a máquina virtual transforma os percursos de maquinagem (designadamente ficheiros no formato *.NCI – *Numeric Control Intermediate*) em código de controlo numérico apto para uma determinada máquina CNC. Esta transformação é realizada pelo pós-processador.

Esta fase é composta inicialmente pela escolha do tipo de máquina:

- Fresadora
- Torno mecânico
- Roteadora
- Máquina de corte por fio

Paralelamente, é necessário escolher qual o ficheiro de controlo (no formato *.control-5), que desempenha a função de base de dados de pós-processadores (ficheiro no formato *.pst), e de centro de definições para os mesmos, e que será associado às definições da máquina escolhida. Estas definições dividem-se de acordo com o tipo de máquina escolhida:

- Definições de fresadora – ficheiro no formato *.mmd-5
- Definições de torno mecânico – ficheiro no formato *.lmd-5
- Definições de roteadora – ficheiro no formato *.rmd-5
- Definições de máquina de corte por fio – ficheiro no formato *.wmd-5

De notar a nomenclatura derivada do inglês: *mmd-mill machine definitions; lmd-lathe machine definitions; rmd-router machine definitions; wmd-wire machine definitions.*

A definição da máquina virtual significa a sua configuração (ver Figura 13).

Do ponto de vista oficinal, e considerando a possibilidade de haver, por exemplo, um armazém de ferramentas no centro de maquinagem, esta configuração inclui a escolha da base de dados da ferramenta a usar no modelo (ficheiros no formato*.tools-5 ou *.tooldb), a base de dados de operações (ficheiro no formato *.operations-5), assim como os seus valores iniciais (ficheiros no formato *.defaults-5), e, por fim, a base de dados de materiais (ficheiros no formato *.materials-5). Esta concentração de escolhas permite, por exemplo, que o utilizador reserve determinado conjunto de operações, ferramentas e materiais, de acordo com as capacidades do seu centro de maquinagem.

Existem neste componente outras opções relevantes, aplicáveis ao modelo global:

- Que informação adicional (comentários e designações) figurará no código de CN?
- Qual é a base do cálculo para as velocidades de corte das ferramentas?
- Como se dá a organização numérica das ferramentas?
- Como é definida a zona de segurança de retração de ferramenta?

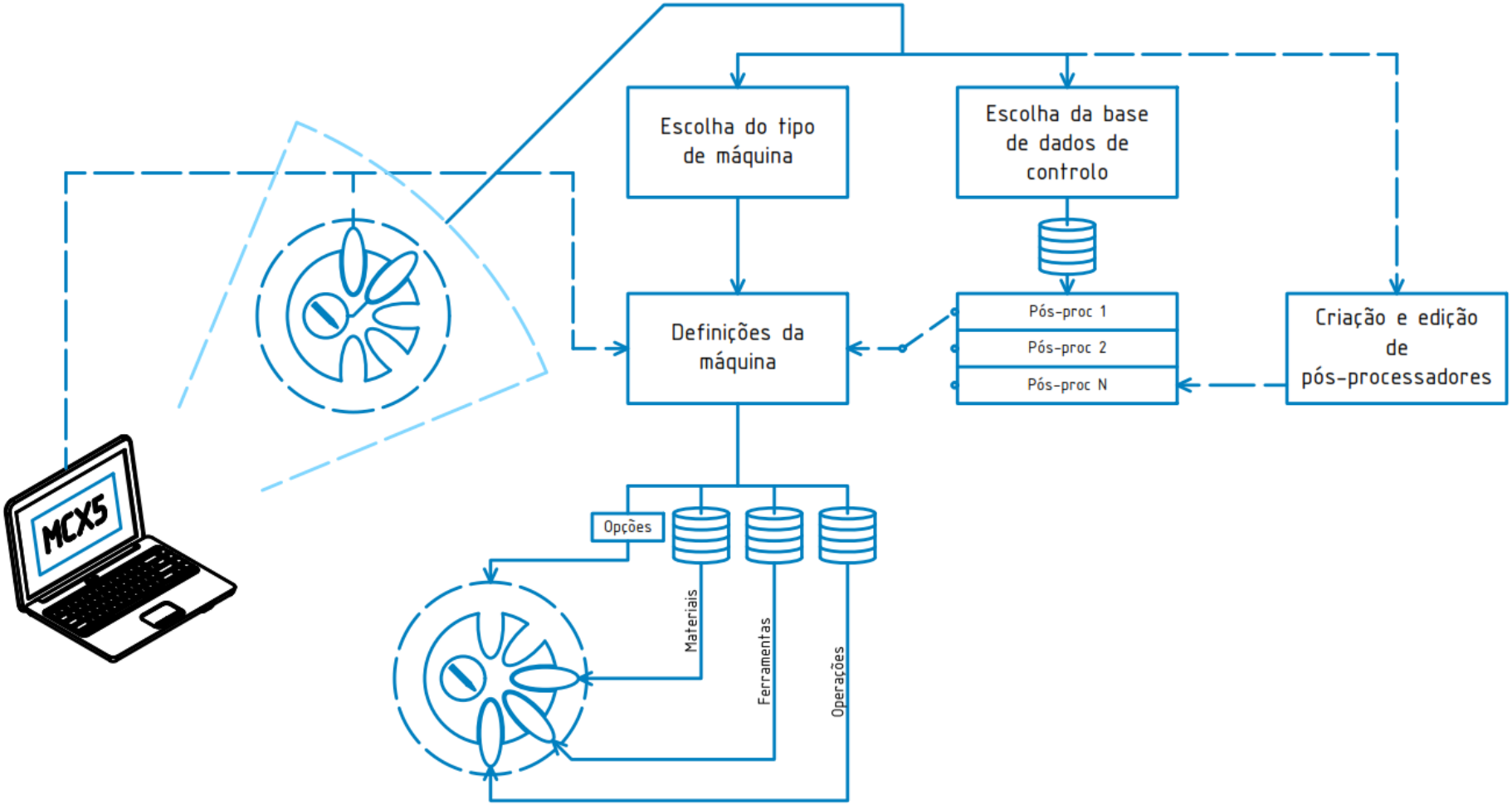


Figura 13 - Componente MÁQUINA; diagrama de definição da máquina.

A especificação do material em bruto – designado em inglês como *stock* – dá-se nesta fase. O *stock* pode ser definido como a forma que o modelo ou peça tem antes de se iniciar a sequência de operações que determinada máquina irá realizar. Esse modelo bruto pode ser associado a geometria já existente no modelo ativo (inclusive criada em paralelo com a geometria final), importada de geometria no formato *.stl, ou gerada por formas primitivas de atravancamento: paralelepípedos ou cilindros (*bounding box*).

A estrutura física da máquina virtual é definida por:

- Parâmetros da cinemática da máquina (velocidades e acelerações máximas)
- Parâmetros operacionais (refrigeração e troca de ferramentas)
- Árvore descritiva da composição da máquina (árvore relacional)
- Integração de dispositivos de fixação na árvore acima mencionada

O Mastercam X5 oferece a possibilidade de configurar a estrutura física da máquina não só através de parâmetros (emulando componentes mecânicos genéricos), mas também através da importação de ficheiros *.stl, ou geometria já existente no modelo ativo. É de mencionar igualmente a possibilidade de integrar a definição de *stock* na estrutura da máquina, tornando-o solidário com os modelos dos componentes mecânicos e de dispositivos de fixação.

As definições do componente máquina são utilizadas nos componentes material, ferramenta, e operações.

O componente responsável pela informação do material define-se como representado na Figura 14. Apesar da base de dados ser definida no componente anterior (máquina), assume-se, com o propósito de melhor distinguir as etapas, que o material é definido num passo distinto. Dessa forma, a sua definição dá-se apenas através da escolha, criação ou edição de um material, que poderá estar ou não presente na base de dados.

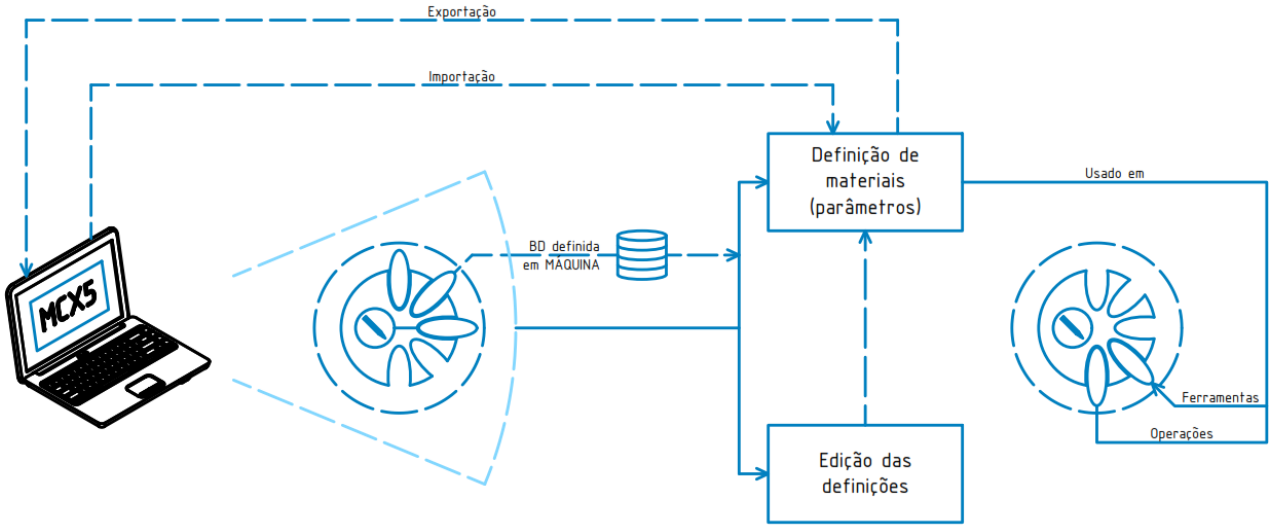


Figura 14 - Componente MATERIAL; diagrama de definição de material.

Os parâmetros associados a um material relacionam o material da ferramenta, o tipo de ferramenta, a natureza da operação, e, naturalmente, o material a maquinar em si. Essa relação baseia-se em percentagens de performance, que ajustam o comportamento de uma variável, face ao valor de outra (i.e.: v.corte broca – Alumínio=100%; Aço=60%).

A gestão de ferramentas baseia-se na escolha da base de dados a usar e opcionalmente na criação, edição, importação ou exportação de ferramentas.

Na Figura 15 representa-se um diagrama representativo desse processo.

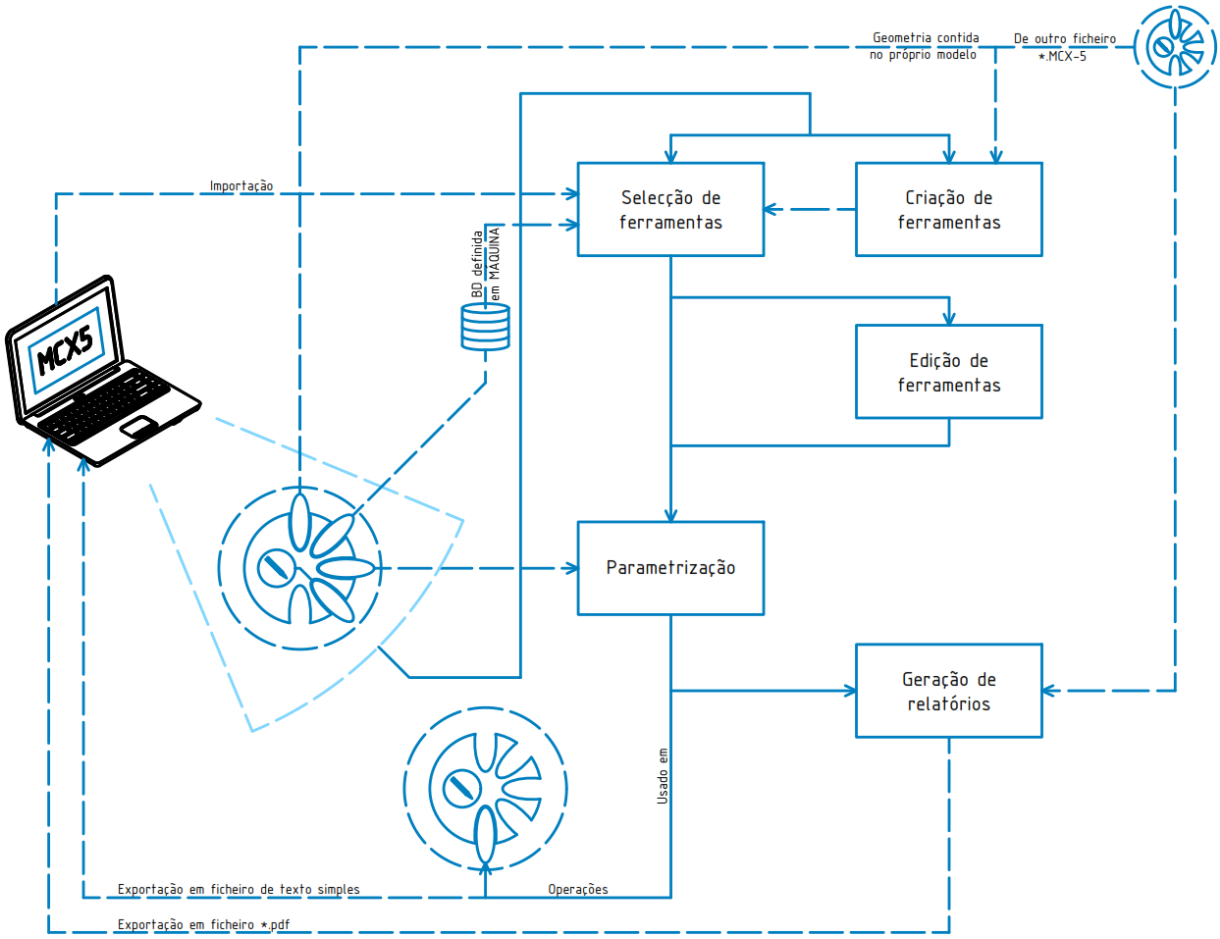


Figura 15 - Componente FERRAMENTA; definição de ferramentas.

Configurados os três primeiros componentes, deve-se configurar as ferramentas. A seleção da base de dados proporciona as seguintes opções:

- O utilizador não quer ter ao seu dispor a gama completa de ferramentas:
  - Nesse caso, deverá selecionar um conjunto de ferramentas representativo das capacidades da máquina ou da disponibilidade do *stock* ferramental da oficina ou ainda do armazém robotizado no centro de maquinagem.
  - Poderá ainda refazer uma base de dados reduzida e específica àquele modelo.
  - Este método tem a vantagem de fornecer, na etapa seguinte, estratégias de maquinagem já adaptadas às capacidades concretas da oficina.
- O utilizador quer ter ao seu dispor toda a gama da base de dados de ferramentas:

- Esta opção garante que o processador de caminhos de maquinagem não está limitado a um conjunto de equipamentos, e poderá fornecer soluções que compensem a longo prazo a aquisição de outras ferramentas.
- Esta opção permite também selecionar um conjunto mais reduzido, assim como criar e alterar ferramentas.

A criação de ferramentas necessita de três conjuntos de informações:

1. Escolha do tipo de ferramenta:
    - É necessário selecionar se se pretende criar, por exemplo, uma broca, uma fresa de topo ou uma ferramenta para um torno, entre outros.
    - Esta criação assenta em standards definidos internamente, conjugando perfis gerais contidos em geometrias *.mcx-5, com informações definidas no próximo passo.
2. Introdução de dados dimensionais:
    - Neste ponto requer-se que o utilizador insira as dimensões que a ferramenta deverá ter, tais como diâmetros, comprimentos, dimensões do *bit* de corte, etc.
3. Especificações de parâmetros de corte gerais:
    - Por fim, deve-se definir os parâmetros associados ao funcionamento da ferramenta – velocidades de corte, número de dentes, ângulo de ataque, material, entre outros.
    - Estes parâmetros serão posteriormente combinados com as informações presentes nas bases de dados de materiais e com os caminhos de maquinagem calculados no componente operações – estes parâmetros passam a ser parâmetros específicos.

Este componente dá ao utilizador a possibilidade de elaborar relatórios automáticos de detalhe reduzido (designação, código da ferramenta e diâmetro/raio de corte), ou de maior detalhe, já com parâmetros de corte e exemplificação do perfil. Esta visualização requer ligação aos ficheiros de perfis gerais das ferramentas (*.mcx-5).

A informação produzida nesta fase é utilizada no componente seguinte – operações – onde serão definidos os dispositivos de fixação da ferramenta

Por fim, reunidos todos os dados dos componentes anteriores, toda a informação se conjuga no componente operações (Figura 16).

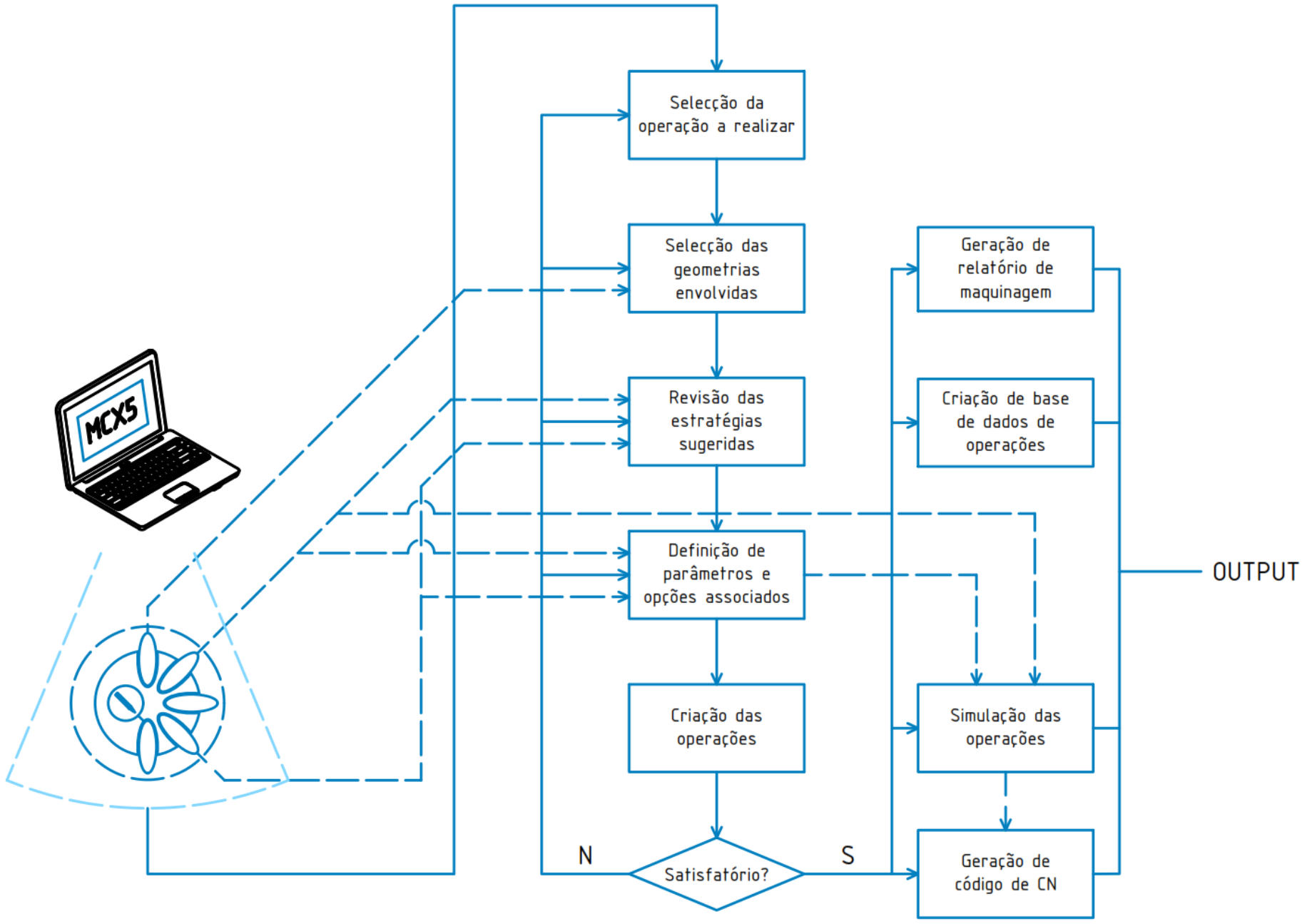


Figura 16 - Componente OPERAÇÕES; definição de operações.

De acordo com a Figura 16, descreve-se o processo de criação de operações de maquinagem:

- A escolha da operação tem o propósito de definir a que conjunto de estratégias respeitará o próximo passo. Desta forma, é possível distinguir precocemente se se trata, por exemplo, de uma furação ou de um desbaste.
- Posteriormente, fazendo uso das geometrias criadas ou importadas no início, selecionam-se linhas, superfícies ou mesmo sólidos que definam curvas, fronteiras ou volumes, e que, por algoritmos de *offsetting* (*offset* – distância que é função da compensação do diâmetro de corte da ferramenta), e algoritmos de varrimento geométrico, servem de base à sugestão de estratégias de maquinagem.
- As definições das máquinas, que contêm as capacidades do centro de maquinagem real, assim como as ferramentas definidas na base de dados, guiarão nesta fase o Mastercam X5. Apresentadas as suas sugestões, cabe ao utilizador empregar o seu conhecimento para discernir qual a estratégia de melhor rendimento, acabamento ou performance, ou ainda que melhor se adapta à geometria (no seu entender).
- A decisão final por uma estratégia abrirá um leque de opções e parâmetros específicos, que, mais uma vez, com critérios do utilizador, devem ser definidos.
- Finalmente, o motor de percursos de maquinagem irá gerar o percurso, qual a ferramenta a utilizar, quais as velocidades, número de passagens, profundidades de corte e distâncias de segurança.

Posto isto, o grau de satisfação, qualitativamente indexado a etapas particulares deste processo, irá mais uma vez responsabilizar o utilizador. O mesmo terá agora de saber se deve alterar algum parâmetro, fazer alguma correção no caminho de maquinagem, adotar uma outra estratégia, ou experimentar ferramentas diferentes. O comparador de geometrias, que contrapõe o que se pretende obter e o resultado final, assim como o simulador do movimento da ferramenta e simulador da máquina virtual, auxiliam esta tomada de decisão.

Por outro lado, caso as operações calculadas sejam satisfatórias, será possível gerar um relatório de maquinagem, gravar as operações numa base de dados, simular o funcionamento da máquina e efetuar deteção de colisões (onde serão utilizadas definições do componente máquina e do componente ferramenta).

Validados todos os aspetos que possam ofuscar a máxima confiança, necessária para que o processo seja comunicado ao centro de maquinagem, é por fim possível obter o *output* deste processo, e gerar o código de controlo numérico (no formato *.NC).

Este capítulo tratou da generalização das capacidades do Mastercam X5, assim como da sua estrutura e modo de funcionamento, transmitindo a forma mais elementar de proceder com o mesmo.

No capítulo seguinte ir-se-ão dar três exemplos, desde a importação de modelos, até à configuração de operações, geração de percursos e obtenção do código de comando numérico.

4 – Exemplos de aplicação do Mastercam X5

No capítulo anterior explanou-se o funcionamento do Mastercam X5.

Neste capítulo, através de três exemplos distintos, mostra-se esse funcionamento assim como uma amostra das suas funcionalidades (particularizado para operações de fresagem). O interesse destes exemplos reside, primeiramente, em validar alguns dos conceitos do estado-da-arte, introduzidos no capítulo 2, assim como provar o conceito de CAM em si como ferramenta digital de apoio à tomada de decisão e planeamento tecnológico. Em segundo lugar, o seu interesse passa por dar a conhecer o Mastercam de uma forma prática, em oposição com o que foi feito no capítulo 3, onde essa abordagem foi mais geral.

O Modelo 1 consiste em mostrar operações 2D de contornos, divididas em patamares. Este exemplo cumpre também a função de introduzir o leitor às tarefas de iniciação de trabalho mais triviais, mas que aqui é pertinente apresentá-las. Este modelo foi escolhido por ser simples o suficiente para ser possível "trazê-lo" para esta dissertação, mas complexo ao ponto do seu planeamento envolver alguns pontos de tomada de decisão.

O Modelo 2 consiste em mostrar operações 3D de desbaste e acabamento, para uma superfície definida por uma *spline*, isto é, NURBS. A escolha deste modelo incidiu essencialmente no facto de constituir um bom exemplo de um planeamento fácil para uma superfície relativamente complexa.

O Modelo 3 consiste em mostrar operações 3D com orientação geométrica radial, em que se fará uma apreciação de vários resultados finais, com um olhar mais crítico para a escolha de determinadas estratégias. Neste caso a escolha deve-se ao facto da sua geometria incluir algumas formas mais difíceis de maquinar, e ter uma orientação radial.

Globalmente, os exemplos conterão 6 etapas distintas:

1. Inicialização do programa e importação de modelo;
2. Escolha da máquina, do ficheiro de controlo, e pós-processador;
3. Escolha e definição das ferramentas de corte;
4. Seleção e configuração das operações;
5. Revisão das operações;
6. Geração de código de comando numérico.

Enquanto que no Modelo 1 as 6 etapas estão todas presentes, nos Modelos 2 e 3 tratar-se apenas das etapas 4 e 5. Isto porque nesses modelos o interesse maior não é tanto nas fases iniciais do trabalho e na fase de geração de código de CN, mas sim nas fases de planeamento e validação.

A análise feita permitirá apontar alguns aspetos de maior relevo, e dar uma noção do tipo de resultados que se obtêm com este *software.* Apesar do esforço em se ilustrar os procedimentos desta análise na sua totalidade, este trabalho não constitui um manual de utilização do Mastercam. Como tal, omitir-se-ão vários pontos da utilização da interface, conforme se entenda necessário.

Para efeitos de avaliação do *software* na qualidade de oferta "chave-na-mão", e para permitir a construção de uma linha de raciocínio que suporte as posteriores alterações ao pós-processador (no capítulo 5), os seguintes elementos serão utilizados sem qualquer alteração prévia:

- Máquina virtual;
- Ficheiro de controlo;
- Pós-processador;
- Material.

Esta avaliação ajuda a perceber o nível em que se encontra este *software* em termos de automatização. Por outro lado, fornece uma medida da qualidade do código gerado num cenário de intervenção mínima por parte do utilizador (no que respeita à configuração do pós-processador).

4.1 – Modelo 1 – Operações de contorno 2D

As operações 2D podem ser consideradas, relativamente às restantes, de utilização trivial. No entanto, por esse mesmo motivo, e devido à sua utilidade, a sua existência é ainda de grande importância no mundo industrial, já que representa o ponto de partida para muitas outras.

Com este exemplo pretende-se dar o exemplo do *workflow* típico da programação da maquinagem de uma peça e colocar, propositadamente, alguns "problemas" derivados das capacidades do *software* e também da geometria em si. Face a esses "problemas", responde-se com soluções para os resolver, seja pela utilização de ferramentas específicas, como pela modificação de algumas opções. No final é gerado o código de CN, destacando linhas mais relevantes, evidenciando as capacidades do pós-processador pré-definido.

Na Figura 17 está representado o modelo 1, através de um desenho de definição simplificado para os devidos efeitos. Note-se as seguintes características:

- 5 patamares distintos (designados de elipse, círculo, losango e quadrado) para movimentos específicos da máquina ferramenta: elíptico, circular, por interpolação, e movimento discretizado para curva *spline*[33];
- Rasgo em cauda-de-andorinha (vista de detalhe B).

[33] Curva *spline*, ou simplesmente *spline*, é um termo associado a um tipo de geometria descritiva em que a curva gerada é definida matematicamente por uma equação polinomial. A sua complexidade aumenta com N. A *spline* é considerada um elemento de NURBS.

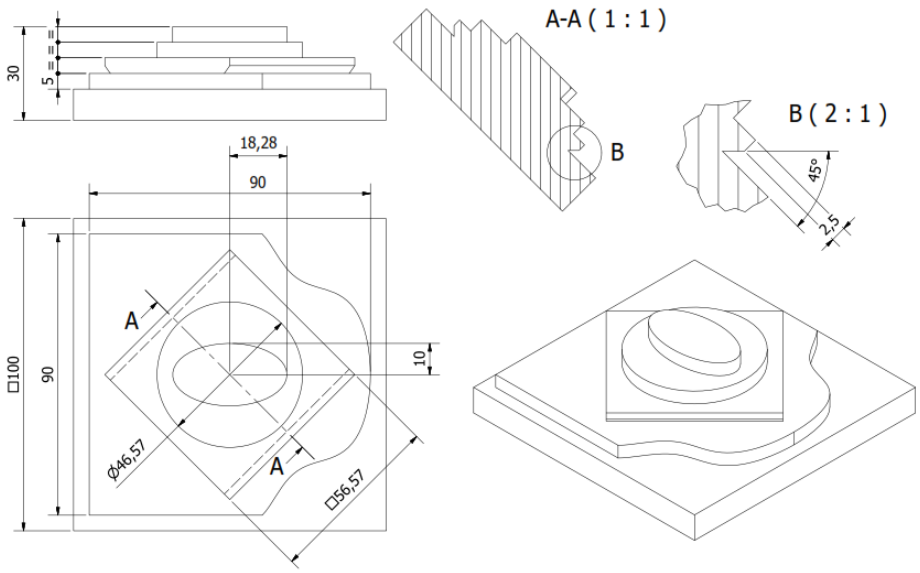


Figura 17 - Modelo 1.

Etapa 1 – Inicialização do programa e importação do modelo

A primeira fase do trabalho consiste na inicialização do programa. Sendo este o primeiro exemplo aproveita-se esta etapa para configurar alguns aspetos da interface, que melhor sirvam o utilizador: disposição dos menus, tamanho dos ícones, cores dos objetos, entre outros. A Figura 18 ilustra o que deverá estar presente na inicialização do programa.

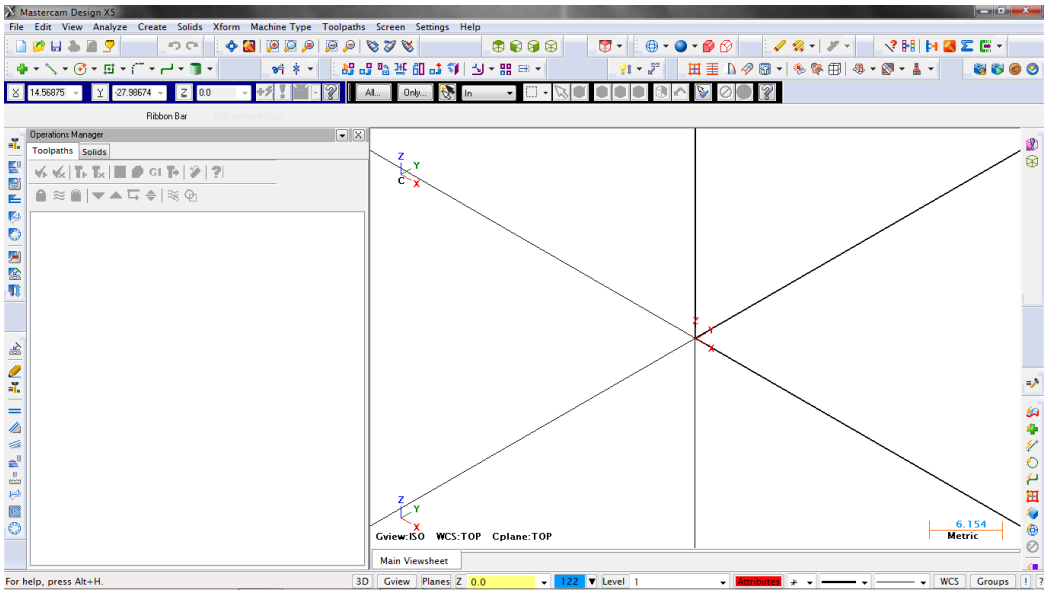


Figura 18 - Inicialização do programa.

Neste exemplo apenas se mostram as funcionalidades CAM. Como tal, não será criado nestes exemplos qualquer modelo com o Mastercam X5, optando-se antes pela importação de modelos construídos com outro *software*, neste caso o Autodesk Inventor. Na Figura 19 pode-se ver o modelo já importado e representado na área de trabalho e na árvore de modelação. O modelo será carregado na posição relativa ao centro do sistema de eixos em que foi desenhado no programa CAD.

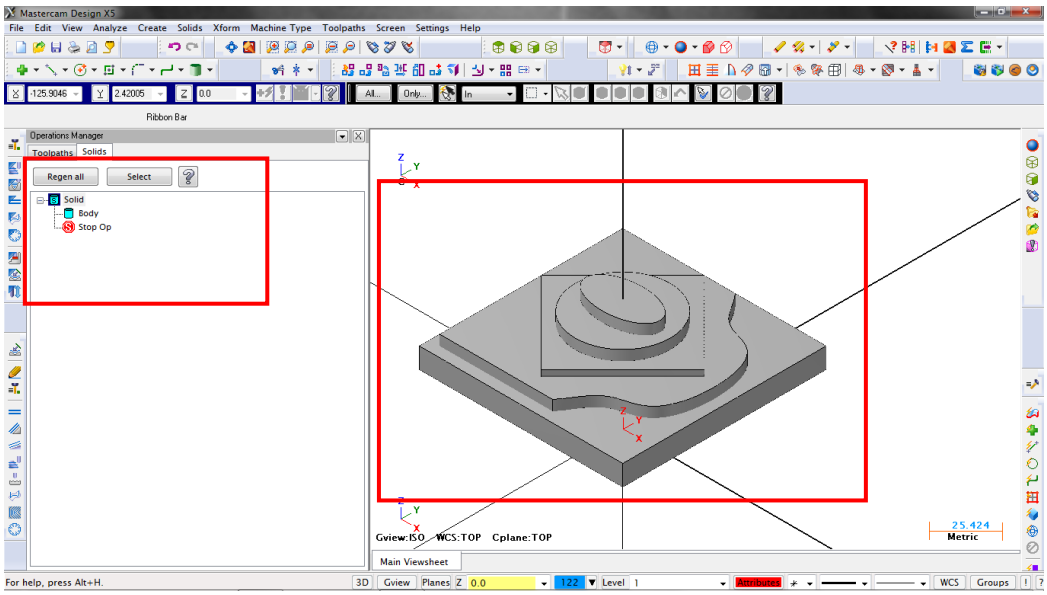


Figura 19 - Importação do modelo.

Etapa 2 – Escolha da máquina, do ficheiro de controlo, e pós-processador

O próximo passo está na escolha de uma máquina virtual (usa-se, como explicado, a máquina pré-definida para 3 eixos do Mastercam), como ilustra a Figura 20.

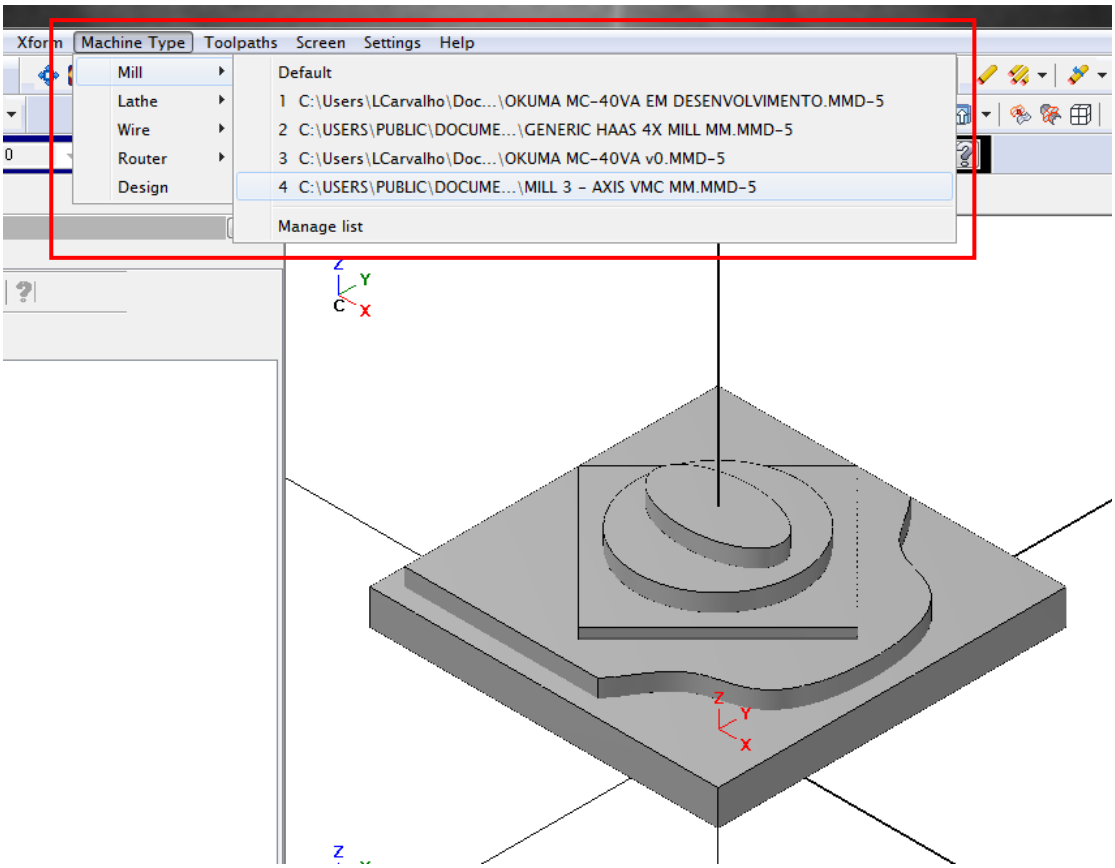


Figura 20 - Escolha da máquina virtual.

A escolha feita é visível na Figura 21. A máquina já se encontra no topo da árvore de operações.

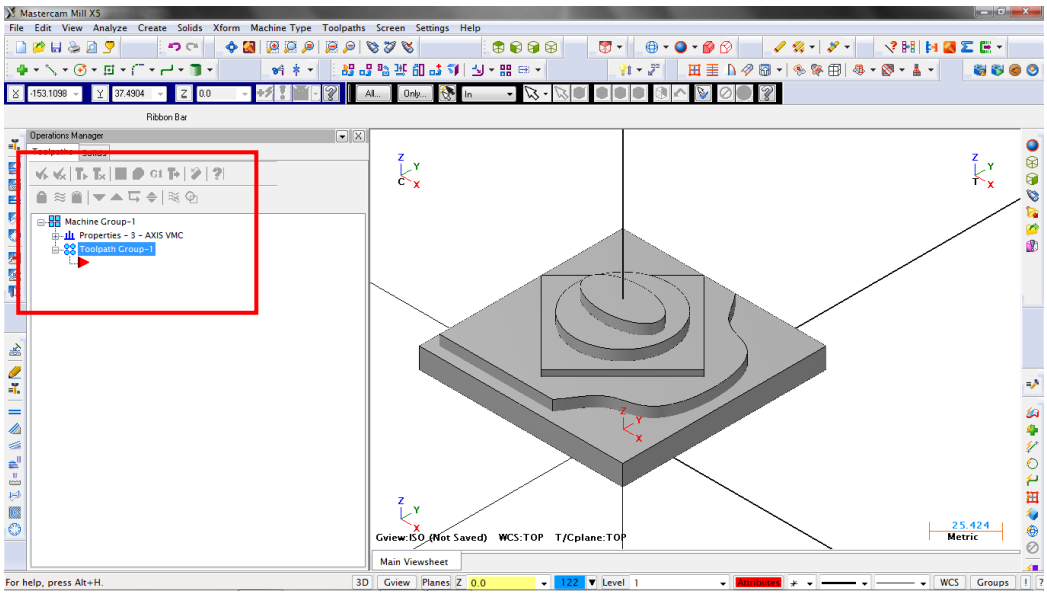


Figura 21 - Observação da máquina virtual na árvore de operações.

Nesta fase é importante definir a forma "bruta", i.e., a forma do bloco de material a maquinar (em inglês *stock).* Esta configuração dá-se nos menus da máquina virtual, e pode ser observada nas Figura 22 eFigura 23. Neste caso, opta-se por um *stock* ajustado aos limites do modelo final (*bounding box* – evidenciada na ilustração pelas linhas a vermelho).

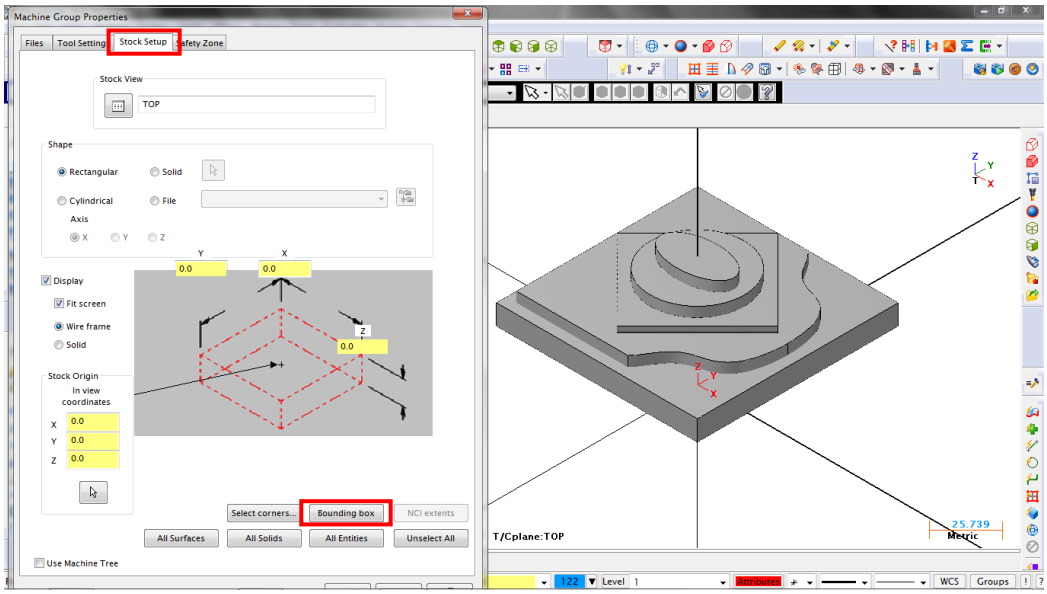


Figura 22 - Definição do *stock* como *bounding box.*

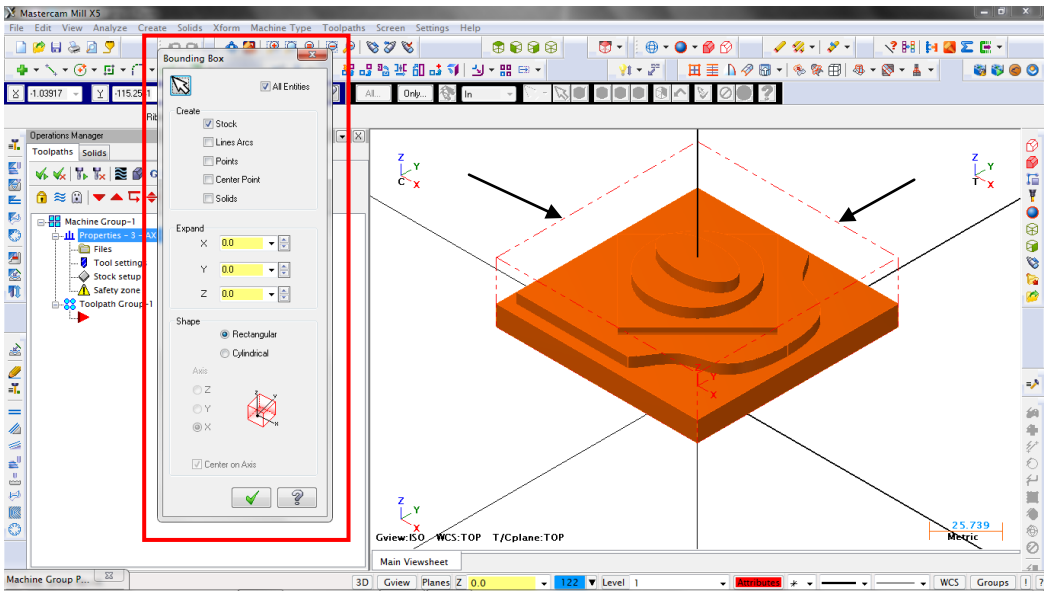


Figura 23 - Outros parâmetros associados ao *stock*.

Etapa 3 – Escolha e definição das ferramentas de corte

Este passo consiste na escolha da base de dados de ferramentas de corte a usar. A base de dados usada nestes exemplos é composta por um conjunto de ferramentas convencionais, mas com uma seleção de tamanhos relevante para as aplicações neste capítulo. Na Figura 24 mostra-se, a partir do menu de configurações gerais da máquina virtual, a escolha da base de dados de ferramentas.

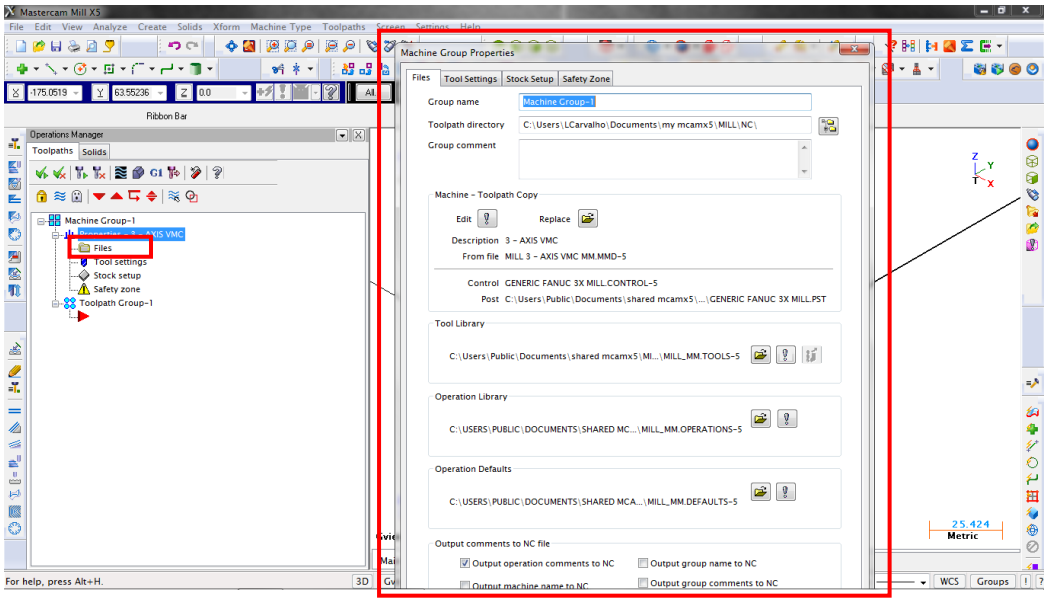


Figura 24 - Menu geral da máquina virtual - escolha da base de dados de ferramentas.

Na Figura 25 observa-se a base de dados escolhida, e o seu registo nas configurações gerais da máquina virtual.

Figura 25 - Base de dados de ferramentas já escolhida.

A escolha desta base de dados serve para consulta aquando da configuração das operações a realizar. Ou seja, esta funcionalidade trata não só de afetar um determinado conjunto de ferramentas a uma máquina, mas também aproximar a sua escolha ao processo de configuração das operações, tornando-a mais expedita. De notar, na figura, o ícone em forma de "pasta". Este ícone sugere, de acordo com a estrutura do *software* já estudada, que a partir deste menu, da máquina virtual, o utilizador aceda às ferramentas por si escolhidas. O que isto significa é que a partir deste ponto existe uma instância da base de dados embebida no modelo *.mcx-5.

Aqui apresenta-se o acesso ao gestor de bases de dados de ferramentas. Na Figura 26 evidencia-se a sua localização na interface.

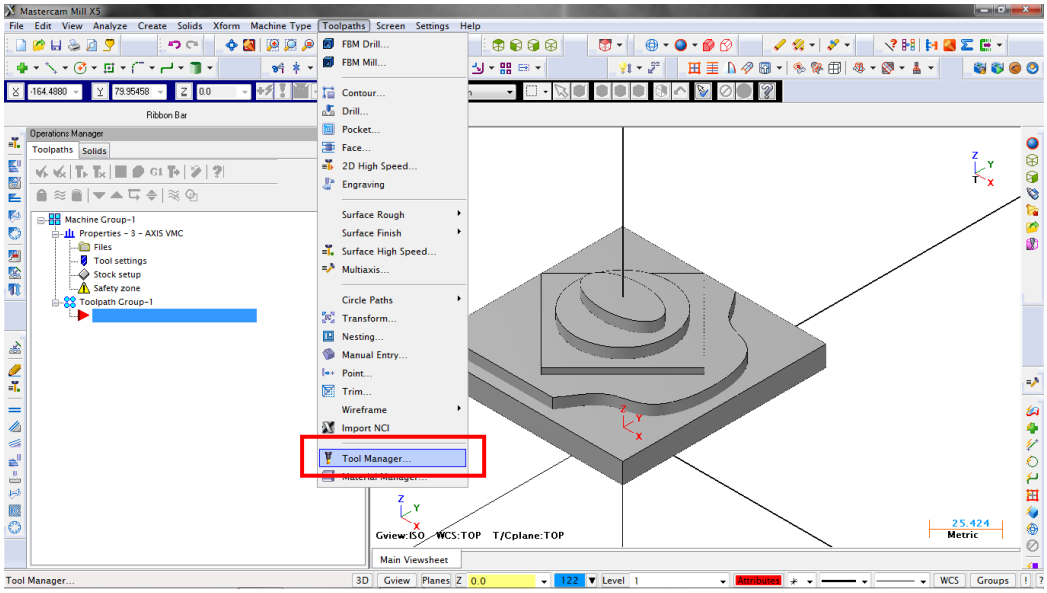


Figura 26 - Entrada no gestor de ferramentas.

O gestor de bases de dados de ferramentas está dividido em duas partes: uma com as ferramentas usadas nas operações – em cima – e outra com todas as ferramentas que a base de dados possui (Figura 27) O que isto implica é que a parte de cima diz respeito a, por exemplo, a afetação dessas ferramentas a um armazém de ferramentas numa máquina, e a parte abaixo representa todas as ferramentas presentes na oficina.

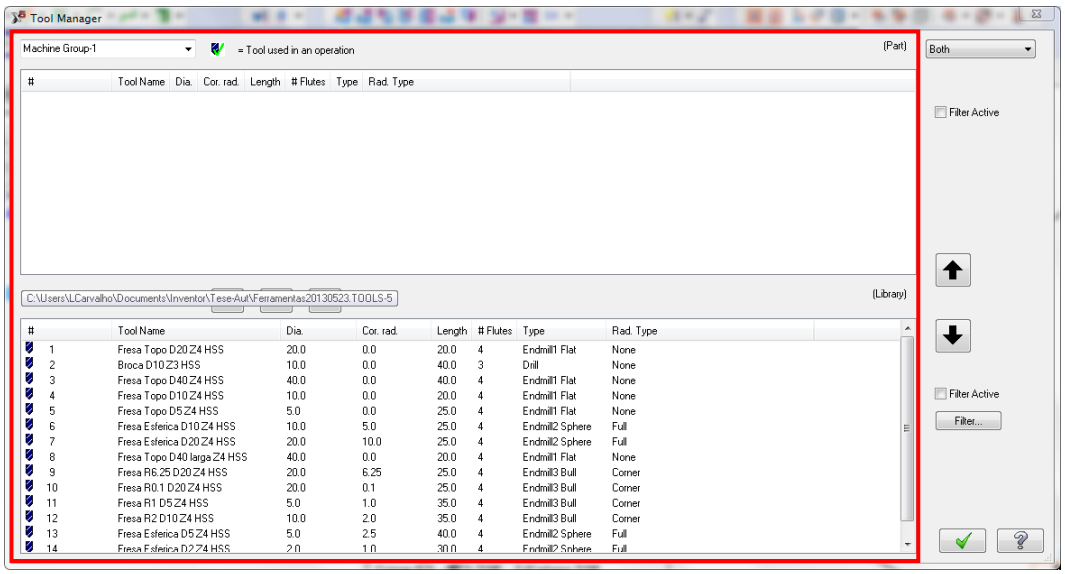


Figura 27 - Gestor de ferramentas.

Etapa 4 – Seleção e configuração das operações de maquinagem

A partir deste momento está-se apto a iniciar o processo de criação de operações. Este começa pela seleção do tipo de operação pretendido. Para este exemplo, escolhe-se operação do tipo *CONTOUR (*Figura 28*).*

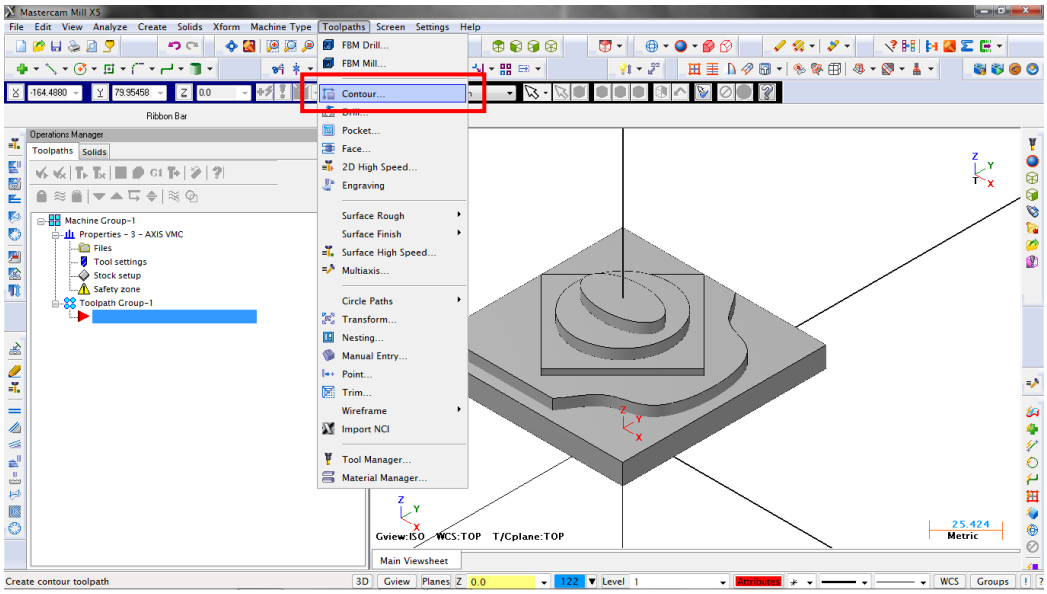


Figura 28 - Seleção da operação de contorno.

Ao fazê-lo, surge o menu de seleção de geometrias. Este menu é altamente flexível no que diz respeito à forma como é possível escolher entidades geométricas, mas peca por ser confuso.

Figura 29 - Seleção de geometrias.

Neste caso usa-se o método de seleção de entidades em sólidos, e serão selecionados os quatro perfis que compõem 4 dos patamares deste modelo. A Figura 30 evidência essa seleção. Embora se pudesse criar uma operação para cada perfil, é mais expedito seleciona-los na mesma operação – não é necessário aplicar condições diferentes a cada perfil.

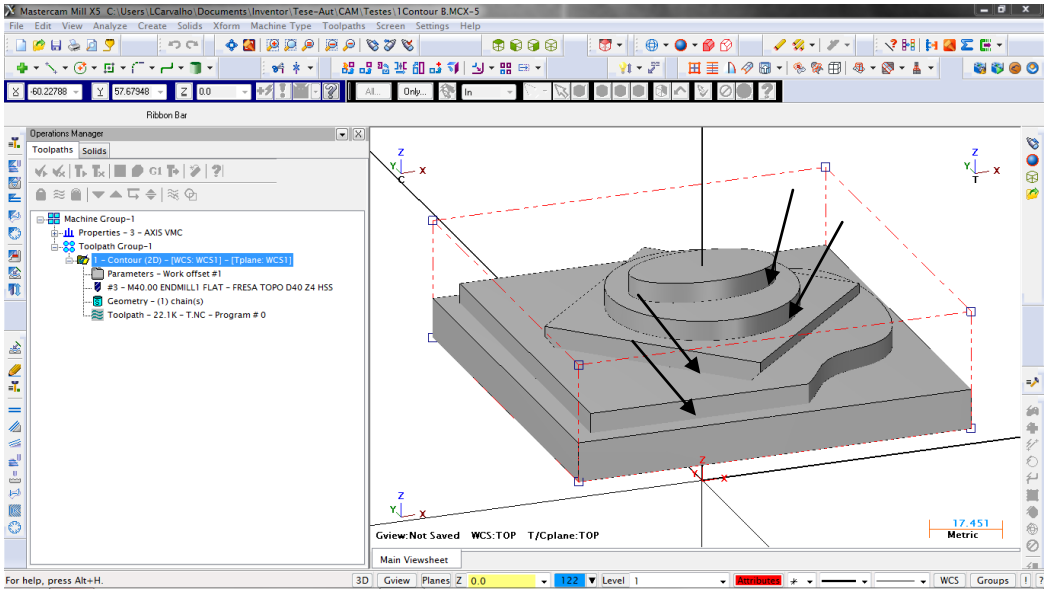

Operations Manager
Toolpaths
Solids
Machine Group-1
Properties - 3 - AXIS VMC
Toolpath Group-1
1 - Contour (2D) - [WCS: WCS1] - [Tplane: WCS1]
Parameters - Work offset #1
#3 - M40.00 ENDMILL1 FLAT - FRESA TOPO D40 Z4 HSS
Geometry - (1) chain(s)
Toolpath - 22.1K - T.NC - Program # 0
Gview:Not Saved WCS:TOP T/Cplane:TOP
Main Viewsheet

Figura 30 - Escolha dos perfis de contorno.

A escolha da operação e posterior seleção das entidades geométricas fazem dispor de seguida um menu com as opções da operação selecionada. Para esta operação as opções que figuram encontram-se listadas à esquerda. Na Figura 31 exemplifica-se a escolha da ferramenta para a operação em causa: uma fresa de topo de 40 mm de diâmetro.

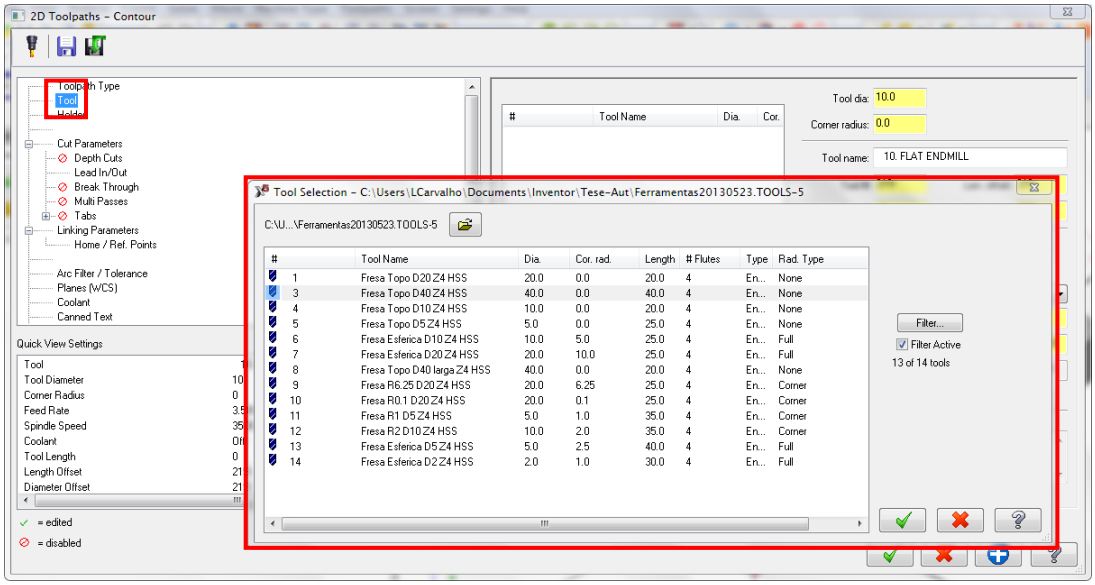


Figura 31 - Escolha da ferramenta para a operação de contorno.

Agora a operação encontra-se criada, e pronta a ser revista. Na Figura 32 são visíveis os 4 percursos gerados: a azul representam-se os movimentos de avanço em corte, e a cor-de-laranja, os de avanço rápido.

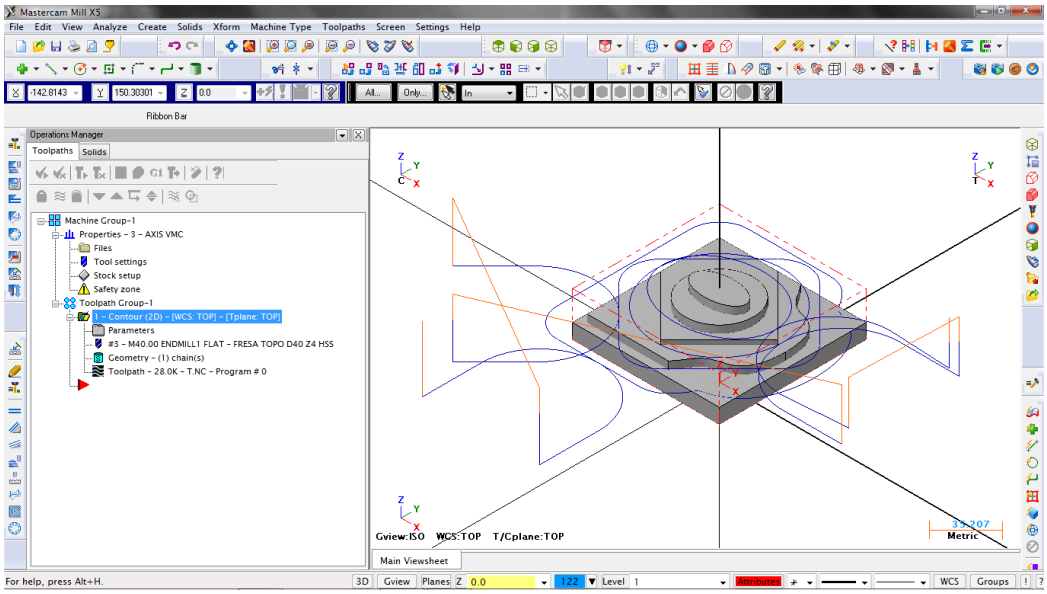

Mastercam Mill X5
File Edit View Analyze Create Solids Xform Machine Type Toolpaths Screen Settings Help
Ribbon Bar
Operations Manager
Toolpaths Solids
Machine Group-1
Properties - 3 - AXIS VMC
Files
Tool settings
Stock setup
Safety zone
Toolpath Group-1
1 - Contour (2D) - [WCS: TOP] - [Tplane: TOP]
Parameters
#3 - M40.00 ENDMILL1 FLAT - FRESA TOPO D40 Z4 HSS
Geometry - (1) chain(s)
Toolpath - 28.0K - T.NC - Program # 0
Gview:ISO WCS:TOP T/Cplane:TOP
Metric
Main Viewsheet
For help, press Alt+H.

Figura 32 - Visualização dos percursos gerados.

Resumindo o que foi feito desde o início até este ponto:

1. Inicializou-se o programa, procedendo à importação do modelo 3D;
2. Escolheu-se uma máquina virtual e configurou-se o *stock;*
3. Escolheu-se a base de dados de ferramentas a usar, revendo-a;
4. Escolheu-se a operação a realizar;
5. Selecionaram-se as geometrias que definem os percursos a gerar;
6. Escolheu-se a ferramenta a usar nesta operação;
7. Geraram-se os percursos.

Como se pode constatar com este exemplo, a ordem dos passos dados corresponde ao explicado no capítulo 3, exibindo a sua lógica, que permite a expedita utilização do Mastercam X5.

Com este exemplo, até a este ponto, conseguiu-se também mostrar – ainda que de uma forma sintética – várias funcionalidades básicas do *software* assim como outras mais avançadas: importação de modelos, gestão de máquinas virtuais, gestão de ferramentas, e geração percursos, entre outras.

A fase seguinte é a da revisão das operações de maquinagem.

Etapa 5 – Revisão das operações

A revisão das operações divide-se em duas tarefas distintas. A primeira consiste em simular a operação (Figura 33) para verificar se não se dão colisões (não ilustrado neste exemplo) e aferir se a qualidade da operação é satisfatória. A segunda consiste em estimar o tempo de ciclo, avaliando se a sua produtividade cumpre os requisitos estabelecidos (aqui apenas se valoriza o ciclo mais rápido, não havendo outro critério).

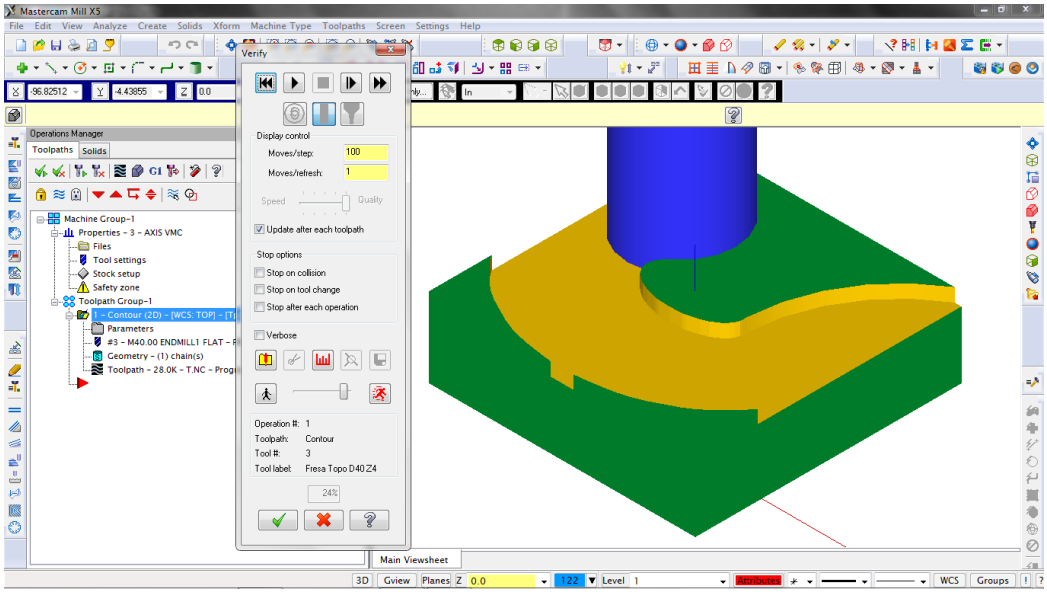


Figura 33 - Simulação de remoção de material em funcionamento.

Aqui observa-se, por exemplo, se, relativamente à forma final é maquinado algum material que não é suposto maquinar. Na Figura 34 observa-se a mesma simulação, agora terminada. Nessa ilustração, a escala de cores auxilia a comparação que será feita entre o modelo final, que se pretende obter, e o modelo maquinado. Cores situadas no meio da escala indicam diferenças mínimas entre os dois modelos. Os extremos acusam material em excesso ou em defeito (a avaliação quantitativa requer calibração).

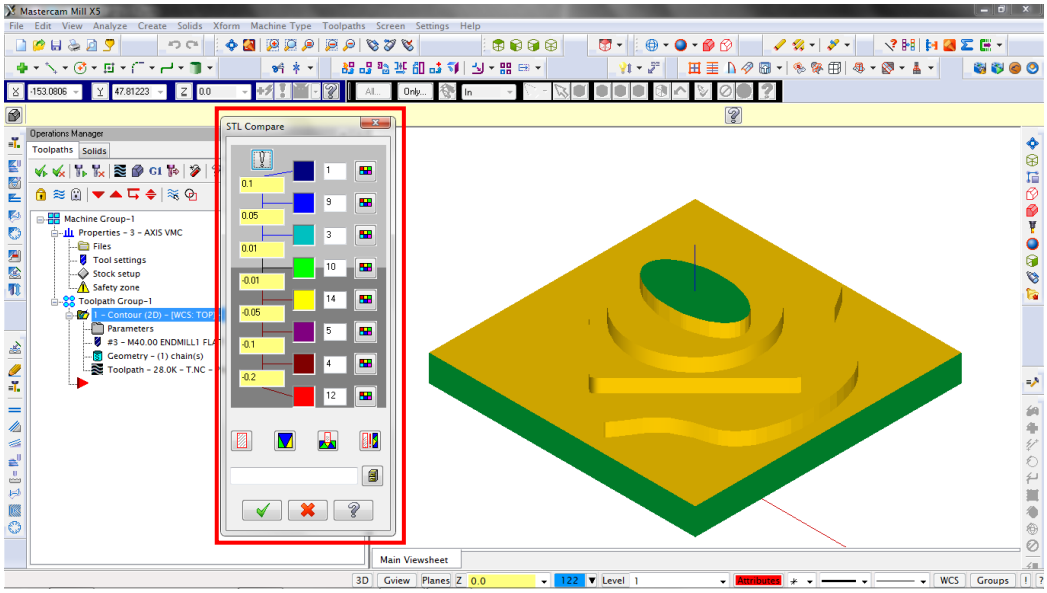


Figura 34 - Simulação de remoção de material terminada.

As Figura 35 eFigura 36 permitem observar alguns problemas no percurso, de acordo com a escala apresentada: enquanto que os patamares elipse círculo, e quadrado apresentam uma qualidade relativamente constante (indicado pelas setas normais), o patamar losango acusa um desbaste exagerado (indicado pela seta a tracejado). A cauda-de-andorinha, definida no modelo (Figura 17, pp. 43), foi eliminada, e os cantos estão maquinados em excesso.

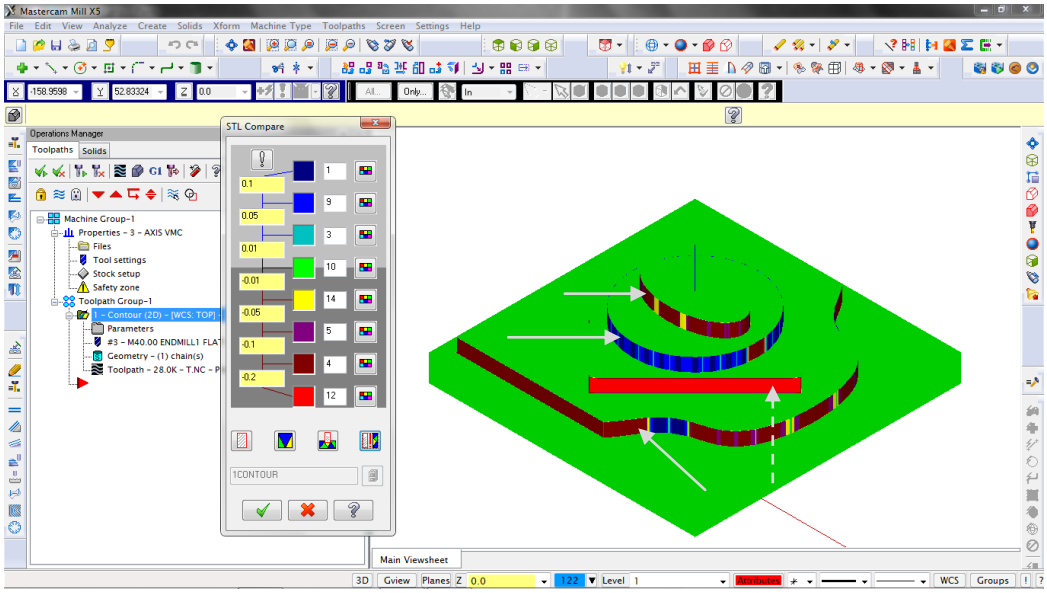


Figura 35 – Simulação da qualidade da maquinagem - destaque de irregularidades.

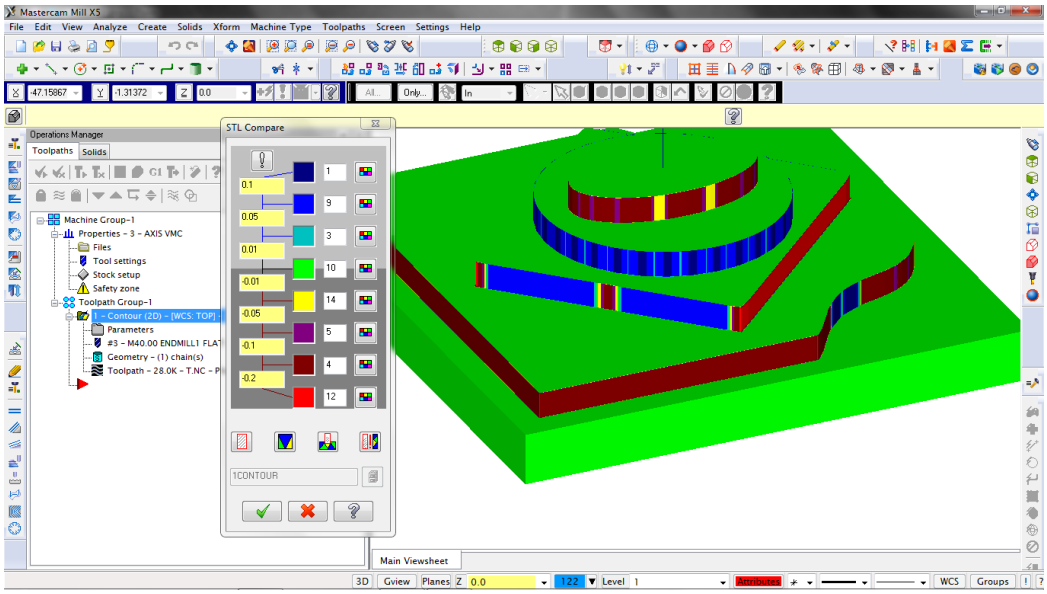


Figura 36 – Simulação da qualidade da maquinagem - melhor vista sobre rasgo de cauda-de-andorinha.

A razão pela qual isto sucede está, no caso do desbaste em excesso, relacionada com a seleção da geometria a maquinar (a geometria escolhida obriga a ferramenta a percorrer um retângulo que passa “dentro” da cavidade da cauda-de-andorinha), e no caso dos cantos desse mesmo patamar, com as definições de contorno de cantos e entrada e saída de ferramentas.

A solução consiste em re-selecionar as geometrias, agora da forma correta, criar uma ferramenta especializada para a maquinagem da cavidade em cauda-de-andorinha e modificar algumas das opções de maquinagem. Na Figura 37, a verde, mostra-se qual a escolha da entidade geométrica que produz o resultado pretendido. A vermelho mostra-se a escolha anterior. Esta escolha faz a ferramenta realizar um quadrado perfeito que define a forma básica desse patamar. Agora a ferramenta especializada já pode desbastar da forma correta.

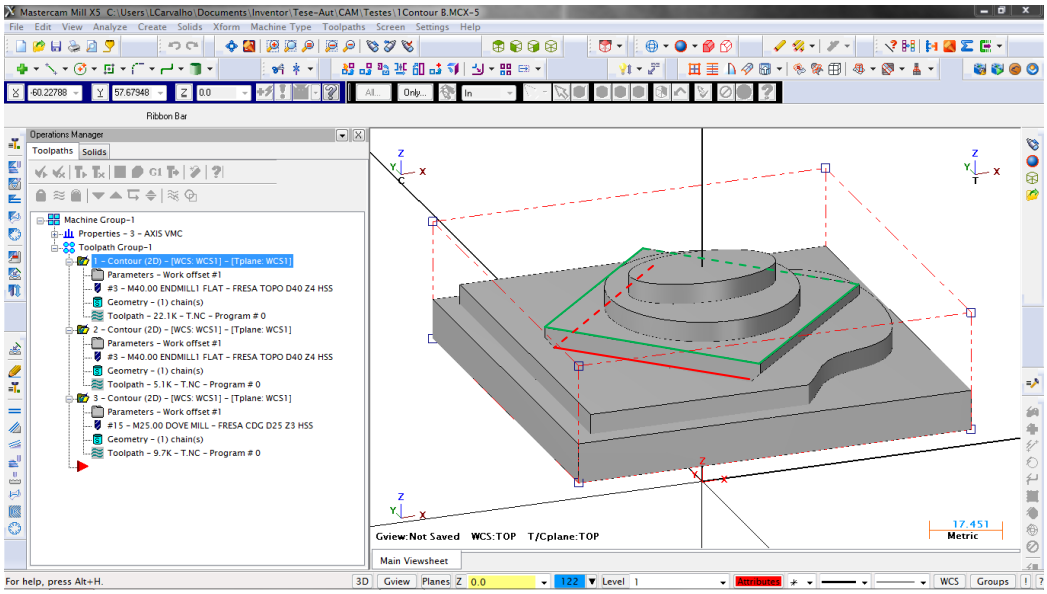


Figura 37 - Re-seleção de geometrias – seleção acertada a verde; seleção errada a vermelho.

Como mostra a Figura 38, a criação de ferramentas (na base de dados global), é feita na gestão de bases de dados de ferramentas.

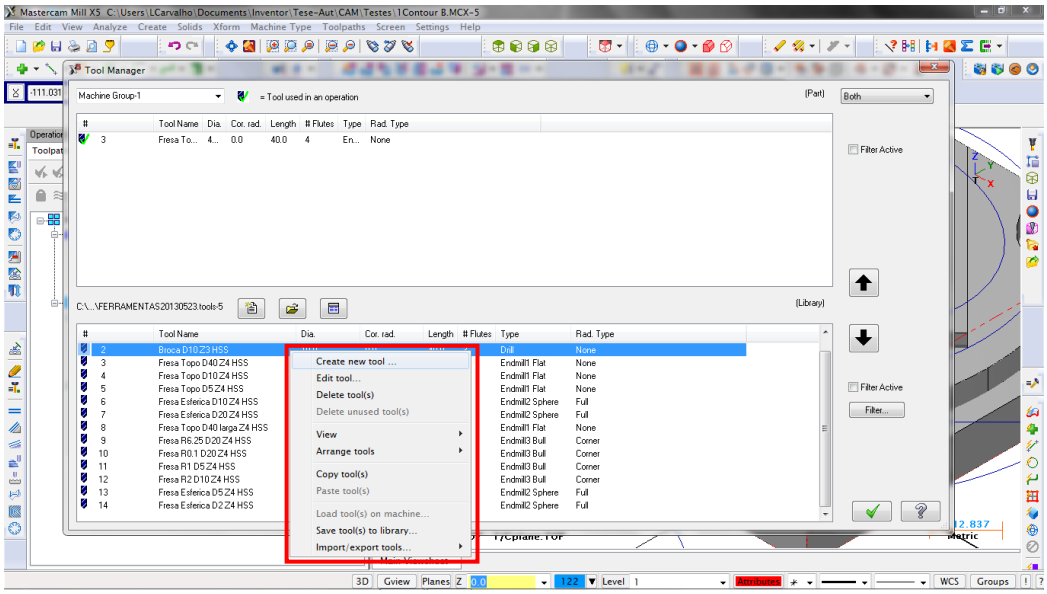


Figura 38 - Início de criação de nova ferramenta.

A definição desta ferramenta fica completa, para os efeitos deste exemplo, com a definição dos parâmetros mais básicos: comprimentos, diâmetros, e algumas definições específicas. Neste caso, por exemplo, temos o ângulo e raio da cauda-de-andorinha. De acordo com o desenho da Figura 17, aqui o ângulo deverá ser 45° e o raio 0. O diâmetro de 25 mm com encabadouro de 12,5 mm garante que a ferramenta consegue desbastar a cavidade totalmente. Fora destas definições apenas se definiu uma designação e um número específico, para que a nova ferramenta possa integrar a base de dados (Figura 39)

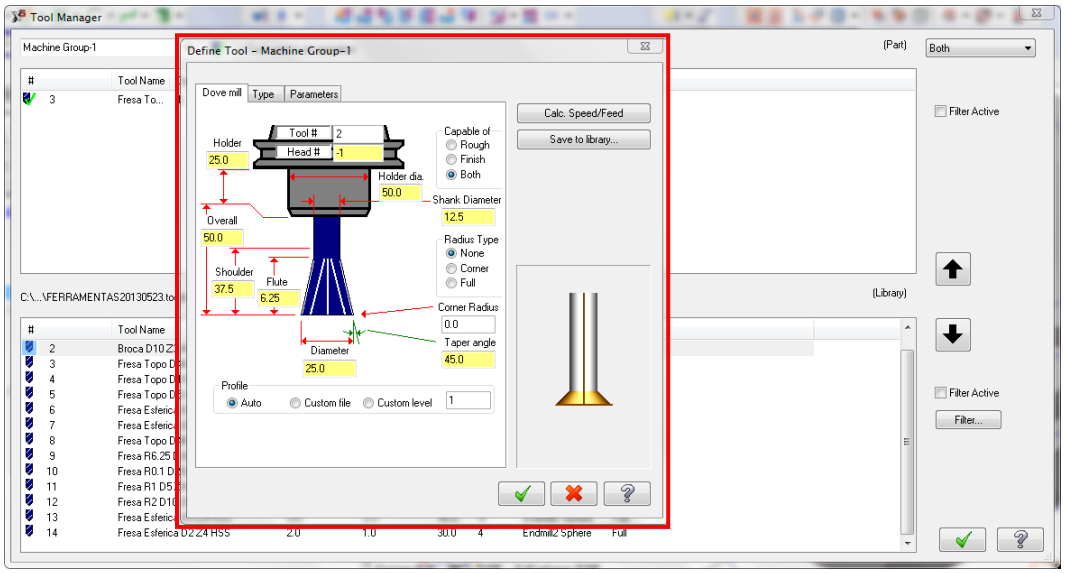


Figura 39 - Parâmetros da fresa em cauda-de-andorinha.

Após esta criação, alteram-se, nos parâmetros da operação, as opções da operação relacionadas com o contorno de arestas vivas (Figura 40). A opção "*Roll cutter around corners"* deve ser desativada. Isto obriga o gerador de percursos a afastar a ferramenta por completo na presença dos cantos nas formas do Modelo 1.

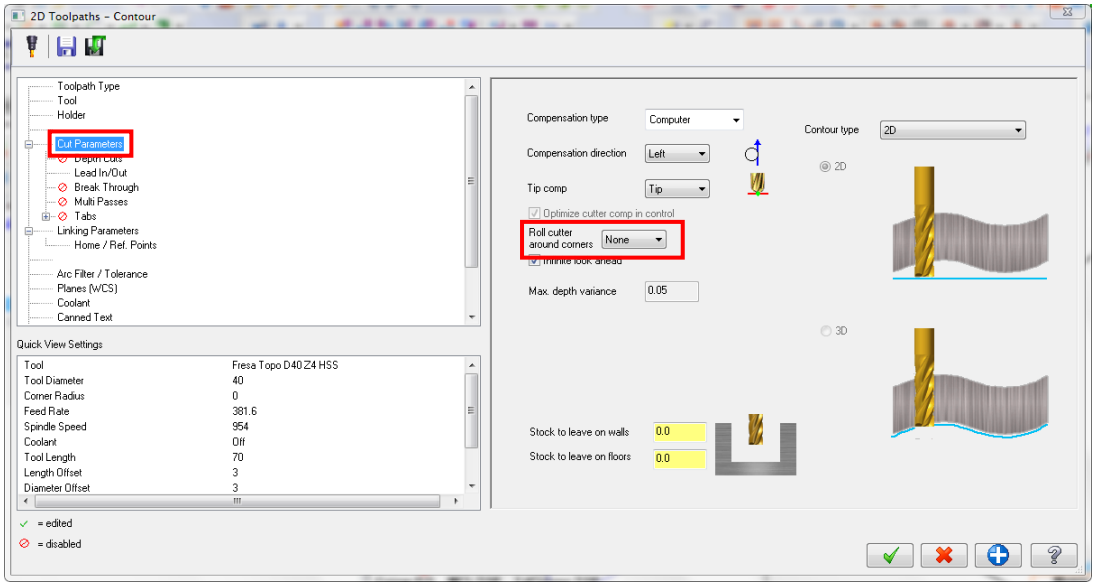


Figura 40 - Alteração de parâmetros de corte - correção dos cantos.

O próximo passo consiste em alterar a forma como a ferramenta entra e sai. É possível alterar essas opções ainda nos parâmetros das operações. Esta modificação previne o corte excessivo de material aquando da sua entrada (Figura 41).

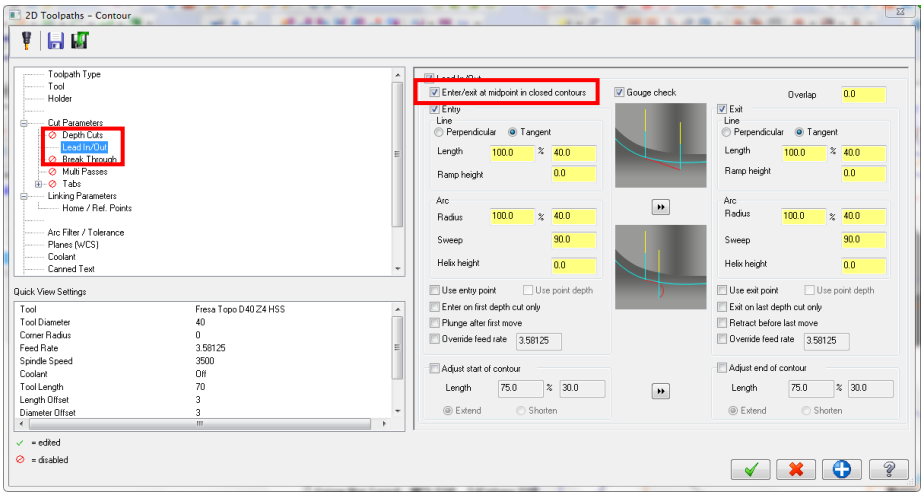


Figura 41 - Alteração de parâmetros da operação - entrada e saída da ferramenta.

A alteração é feita desativando a opção "*Entry/exit at midpoint in closed contours*". Isto obriga a ferramenta a entrar e sair nos cantos dos contornos, fazendo com que o primeiro contacto seja já em pleno movimento de corte. Após esta alteração, é necessário dividir o percurso em 3 partes: uma que define a maquinagem dos patamares círculo, elipse e quadrado; uma que define o losango; uma final que faz a abertura da cauda-de-andorinha. A razão pela qual é necessário dividir o percurso em 3 partes é evidenciada na Figura 42. A seleção das entidades geométricas para maquinar os patamares círculo, elipse e quadrado assenta numa curva complanar com o plano-limite inferior de corte, i.e., com a ponta da ferramenta. Devido às alterações na seleção da geometria para o patamar losango, a curva que agora define o movimento da ferramenta não é complanar com o plano-limite inferior de corte desse patamar (Figura 42, seta vermelha). Logo, deve ser feita uma correção na operação, ajustando-a.

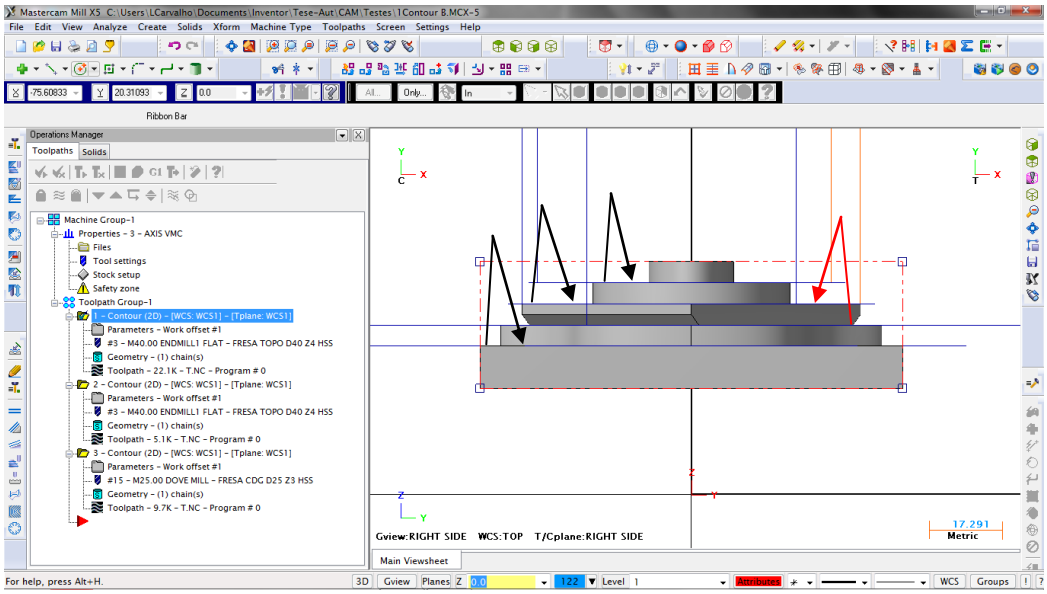


Figura 42 - Explicação do problema de seleção de patamares e dos ajustes das profundidades de corte.

Essa correção, quando aplicada, é vinculada a todo o percurso da operação em causa. Será então necessário separar os percursos, isolando a correção para o percurso de maquinagem do patamar losango. Da forma inicial, os outros percursos já gerados e que estão bem definidos iriam ficar 5 mm abaixo da cota esperada. Na Figura 43 está representada a opção responsável por essa correção, e o local onde é aplicada.

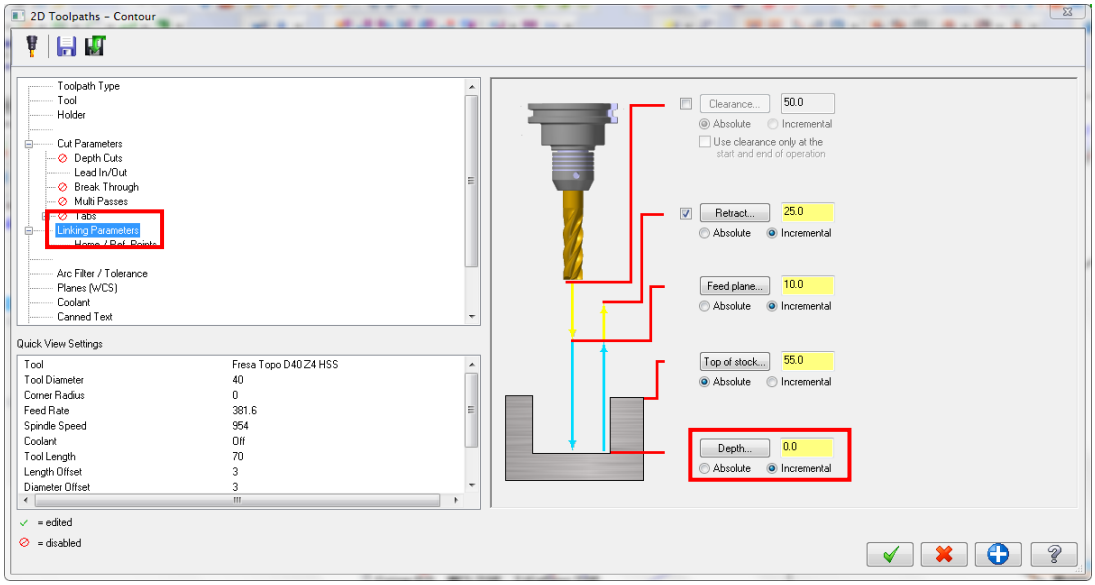


Figura 43 - Alteração da profundidade de corte – alteração do valor de 0 para -5 mm.

Por fim, é necessário selecionar as entidades geométricas que definem o percurso da fresa cauda-de-andorinha (Figura 44, a verde). O processo deve, naturalmente, ser repetido do lado oposto, embora após a sua preparação, possa apenas ser espelhado[34].

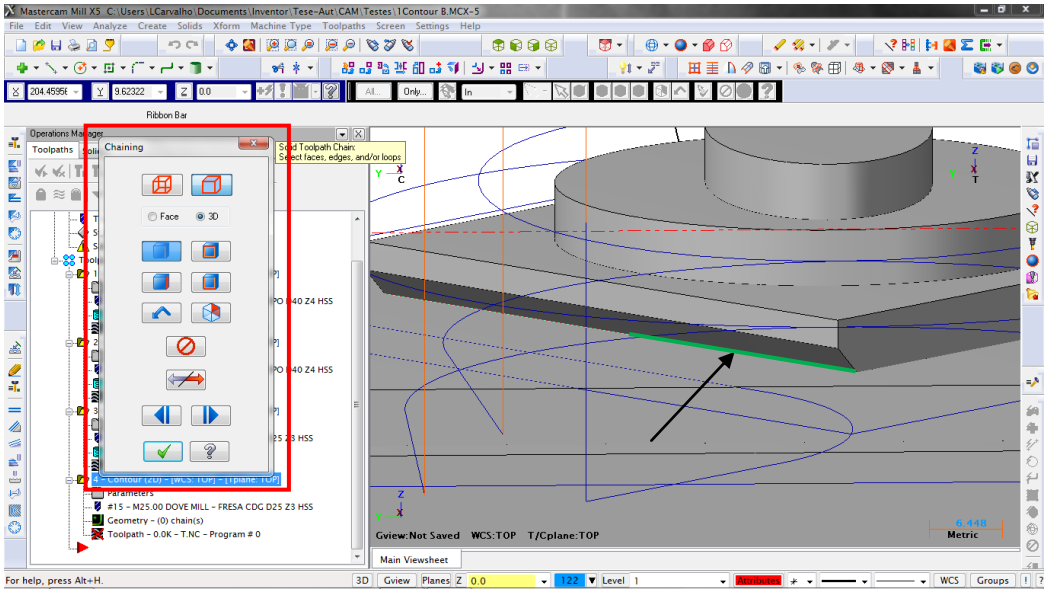


Figura 44 - Seleção da aresta que guiará a ferramenta.

[34] Em seleções simples como neste caso, tal procedimento não acarreta uma redução de tempo significativa.

A Figura 45 representa uma vista de topo do modelo com todos os percursos de maquinagem definidos. É possível distingui-los: 1 – elipse; 2 – círculo; 3.1 e 3.2 – losango; 4 – quadrado. Aqui notam-se as novas entradas e saídas do patamar losango.

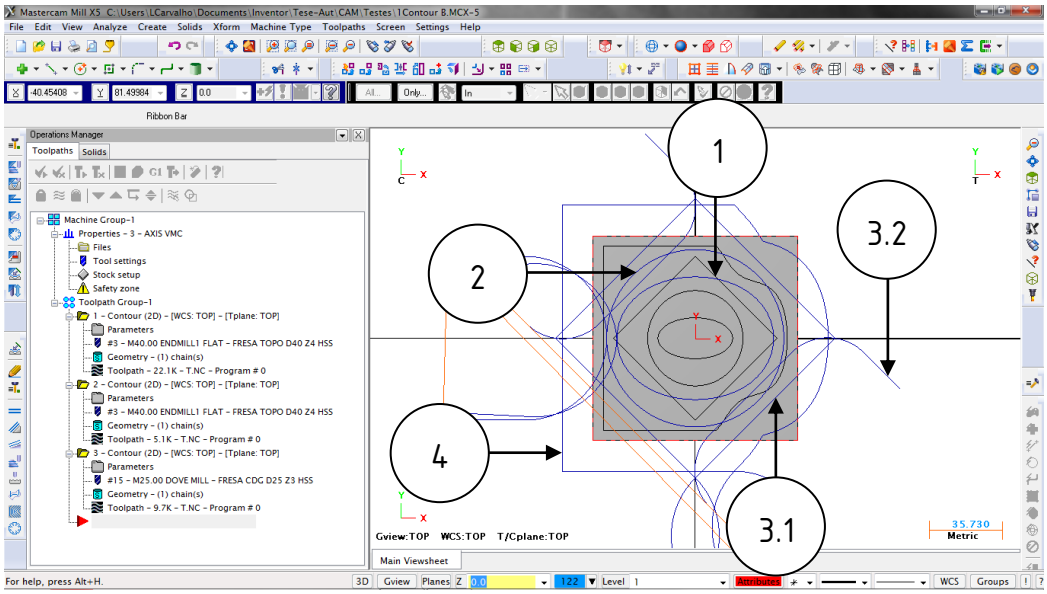


Figura 45 - Revisão dos percursos gerados.

O resultado da simulação com o novo percurso, como mostra a Figura 46, recompensa os esforços de se uniformizar a qualidade da maquinagem. Com a exceção de algumas irregularidades na entrada e saída da cavidade em cauda-de-andorinha – solucionáveis ajustando parâmetros de entrada e saída – todos os patamares apresentam acabamento uniforme[35].

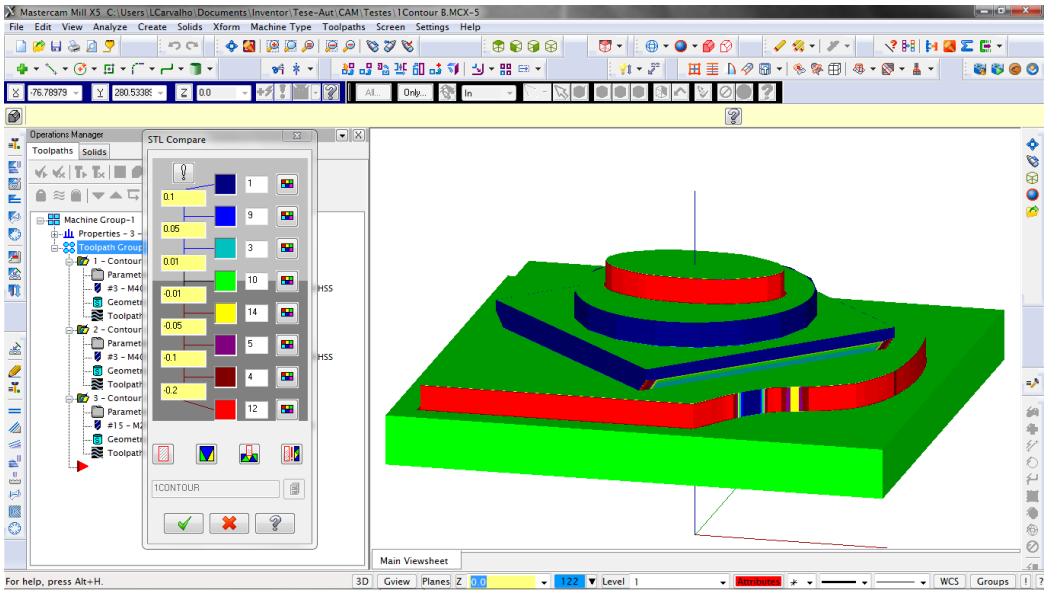


Figura 46 - Verificação do resultado final.

[35] Numa primeira abordagem, é mais interessante procurar uniformidade. A uniformidade indica que o processo está a acompanhar bem a geometria definida. Posteriormente é possível melhorar a qualidade, por exemplo, reduzindo as tolerâncias nos parâmetros da operação.

Etapa 6 – Geração de código de comando numérico

Estando o modelo validado, procede-se agora à geração do programa de comando numérico. Esta fase consiste na definição de um sistema de eixos relacionado com a peça, para que na máquina ferramenta, o operador possa referenciar a máquina a um ponto do bloco a maquinar. Na Figura 47 ilustra-se o início deste processo, com a entrada no menu da gestão de sistema de eixos.

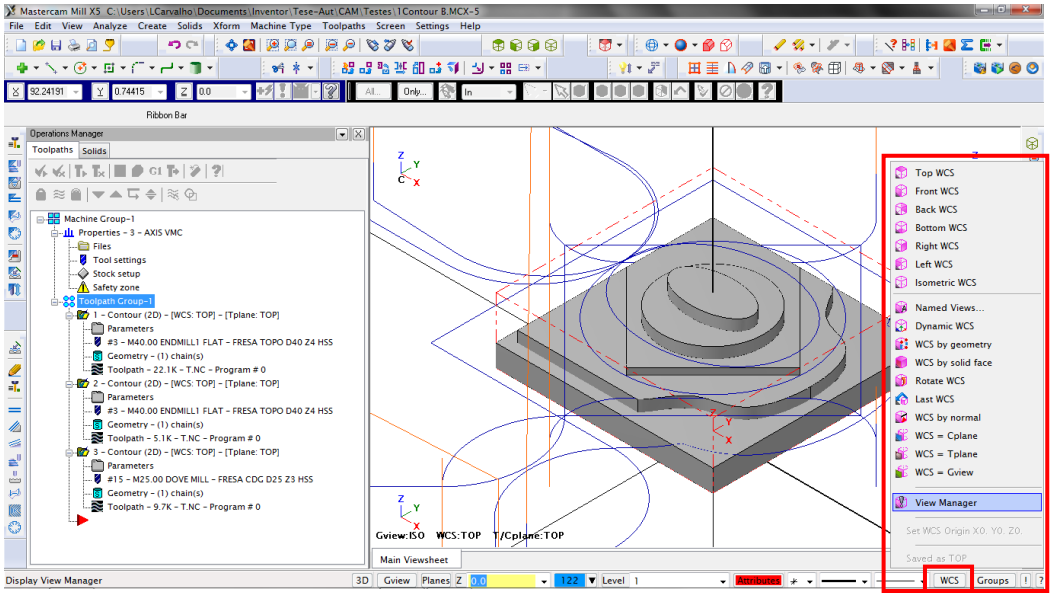


Figura 47 - Iniciação da escolha do sistema de coordenadas.

De seguida, cria-se um sistema, definindo para o efeito um "zero" num dos cantos do *stock*, e posteriormente, a orientação do sistema (Figura 48,Figura 49 eFigura 50). Feita a definição do sistema de eixos, deve-se indicar qual o número a que corresponde esse sistema, para ser usado em conjunto com a listagem dos sistemas de eixos do ficheiro de controlo, e do controlador da máquina. Neste caso, aplicou-se o *work offset #1* – sistema de eixos 1.

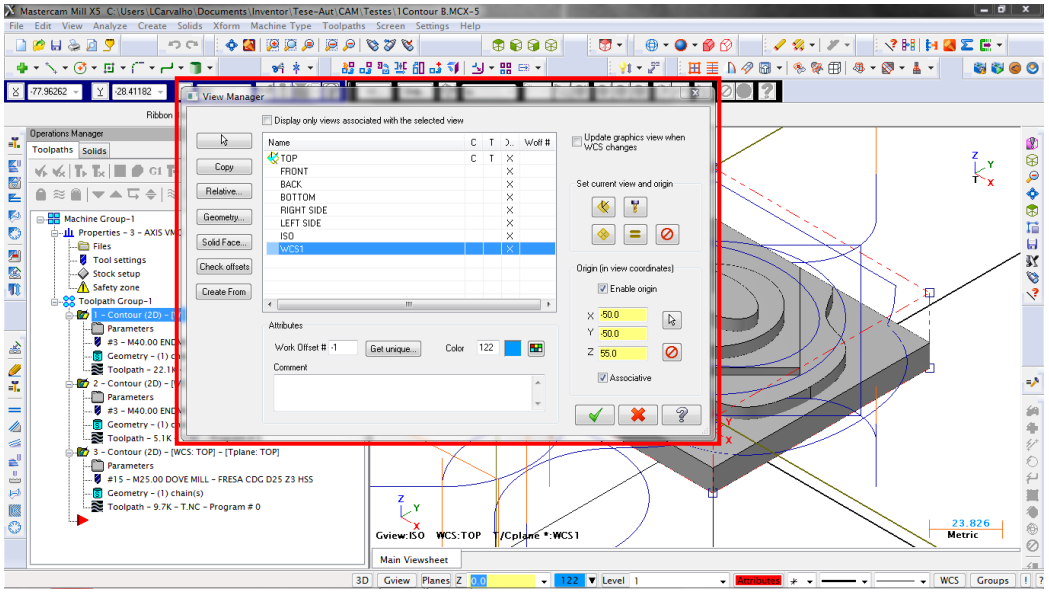


Figura 48 - Menu de definição de sistemas de coordenadas – criação do sistema de coordenadas WCS1.

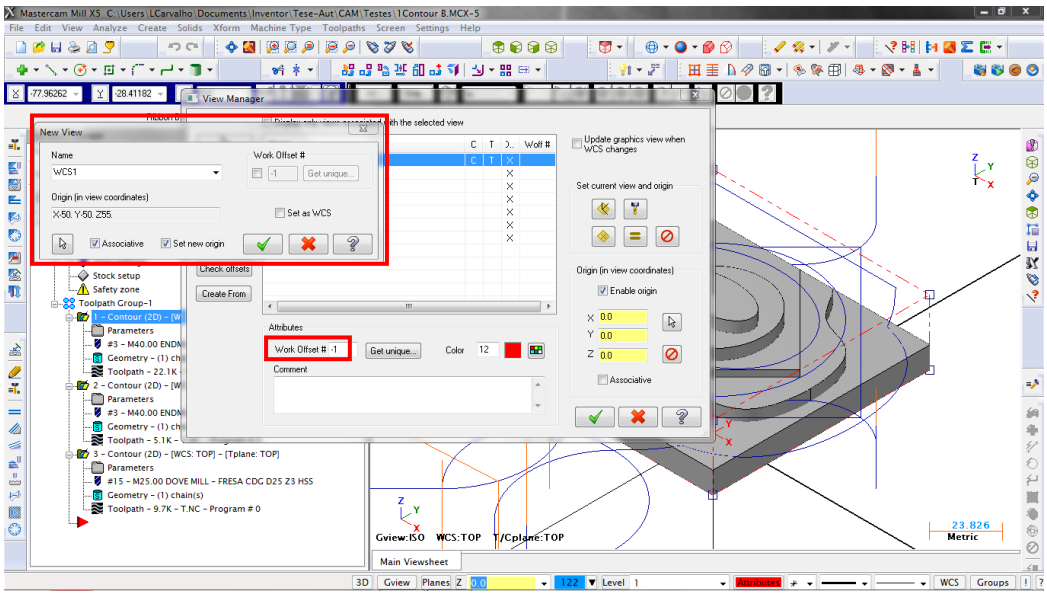


Figura 49 - Revisão do nome do sistema de coordenadas.

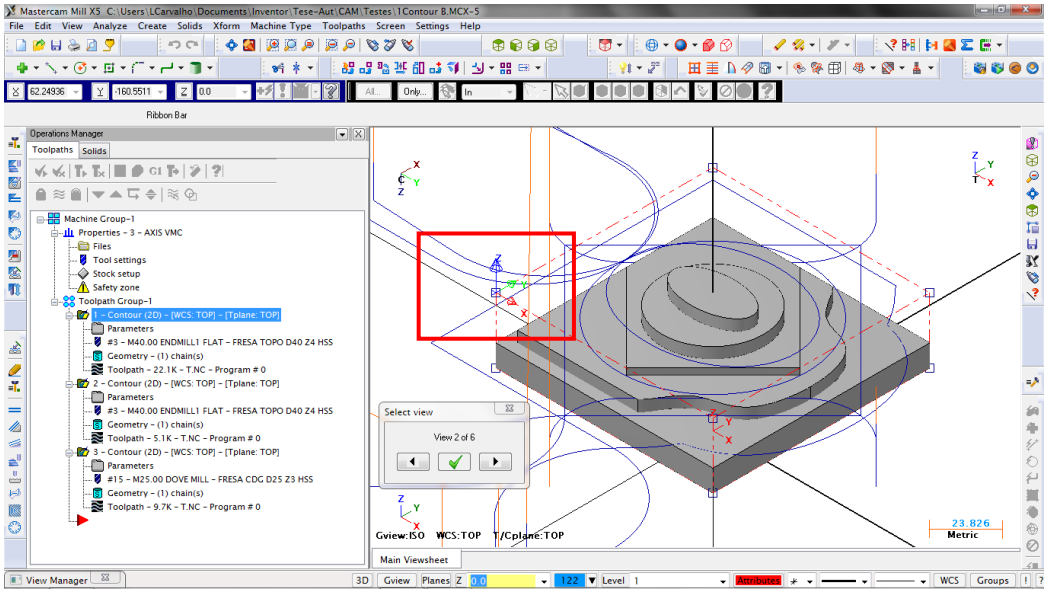


Figura 50 - Sistema de eixos associado à peça.

Definido o sistema de eixos, deve-se agora configurar as operações para que as mesmas se redefinam de acordo com o novo sistema de eixos.

Este processo consiste em definir, nos parâmetros de todas as operações, a existência de um sistema de eixos de trabalho, e qual as suas coordenadas (Figura 51).

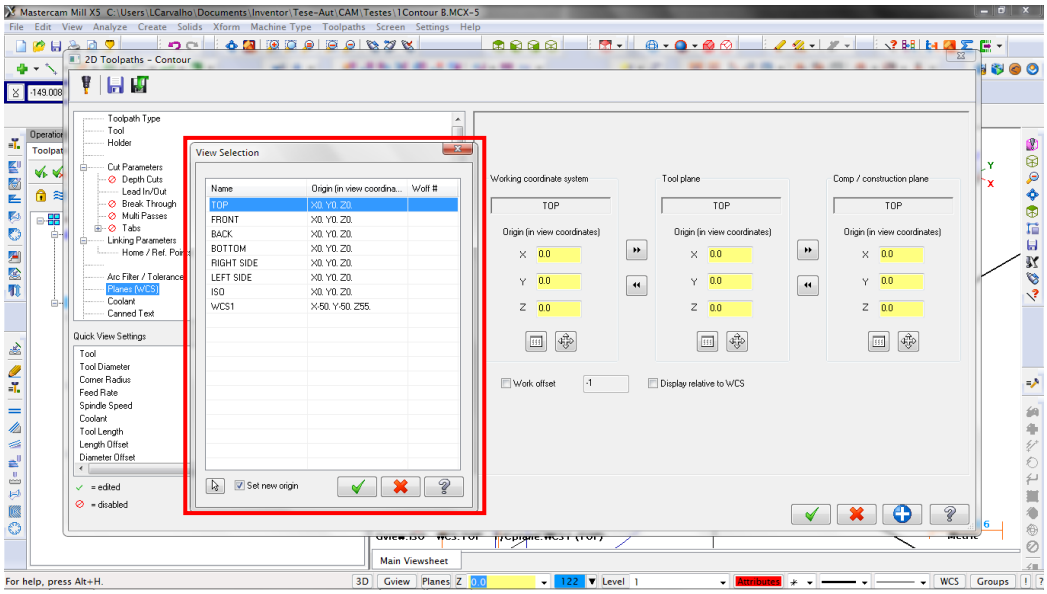


Figura 51 - Aplicação do sistema de coordenadas às operações do Modelo 1.

Os percursos gerados, assim como a definição do sistema de eixos a usar nestas operações, estão finalmente definidos e validados. Resta agora o seu processamento e pós-processamento para que se obtenha o código de CN.

Por ser demasiado extenso, reduz-se o código de CN ao seguinte excerto, comentando algumas particularidades que se entendem relevantes:

```
%
O0000(MODELO1)
(DATE=DD-MM-YY - 01-07-13 TIME=HH:MM - 16:43)
(MCX FILE - C:\USERS\LCARVALHO\DOCUMENTS\INVENTOR\TESE-
AUT\CAM\TESTES\1CONTOUR B.MCX-5)
(NC FILE - C:\USERS\LCARVALHO\DESKTOP\MODELO1.NC)
(MATERIAL - ALUMINUM MM - 2024)
( T3 | FRESA TOPO D40 Z4 HSS | H3 )
( T15 | FRESA CDG D25 Z3 HSS | H15 )
```

(Até este ponto é feito o cabeçalho do programa. Esta parte do programa consiste na apresentação de informações como a data e a hora, localizações de ficheiros e listagem de ferramentas. Seguidamente dá-se o início da operação.)

```
N100 G21
N102 G0 G17 G40 G49 G80 G90
```

(G49 - Cancelamento do *offset* do comprimento da ferramenta.)

```
N104 T3 M6
```

(Chamamento da ferramenta T3, e colocação no *spindle*, com a função auxiliar M6.)

```
N106 G0 G90 G55 X-70.189 Y13.644 S954 M3
```

(Sistema de eixos definido - G55. Derivado da definição de *work offset = 1.* Neste ponto dá-se o primeiro movimento, e ativação do *spindle*, com a função auxiliar M3.)

```
N108 G43 H3 Z80.
```

(G43 - Definição do *offset* do comprimento da ferramenta, posição de memória H3. A partir deste ponto a máquina descreverá o percurso planeado no Mastercam.)

```
N110 Z65.
```

```
(…)
N402 G0 Z80.
N404 M5
N406 G91 G28 Z0.
N408 M01
```
(Aqui é dada ordem de paragem do *spindle* (M5) e posteriormente a interrupção do programa (M01), para que depois se possa efetuar a troca da ferramenta.)
```
N410 T15 M6
```
(Com a instrução T15 M6, ordena-se a troca de ferramenta. Neste ponto a ferramenta T3 será colocada de volta no armazém de ferramentas).
```
N412 G0 G90 G55 X150.104 Y25.251 S1527 M3
N414 G43 H15 Z80.
```
(Definição do *offset* do comprimento da ferramenta, posição de memória H15)
```
N416 Z65.
(…)
N450 G0 Z80.
N452 M5
N454 G91 G28 Z0.
N456 G28 X0. Y0.
N458 M30
%
```
(Retorno à posição de origem (G28) e fim do programa (M30).)

Terminado o exemplo, faz-se um resumo do que foi realizado neste último ponto:

1. Simulou-se a operação, verificando a existência de algumas irregularidades;
2. Apoiado pela visualização das irregularidades, foi possível tomar uma decisão relativamente às soluções para os problemas encontrados;
3. Aplicaram-se as modificações necessárias;
4. Validaram-se os percursos gerados.

Com este resumo pretende-se destacar que se exercitou, nestas 6 etapas, um caso típico de apoio à tomada de decisão e ao planeamento tecnológico, aplicado a um caso específico.

Para além das linhas comentadas no código apresentado com o Modelo 1, realçam-se as seguintes linhas:

- Linha N100: escolha do sistema métrico (G21)
- Linha N102: escolha do modo de avanço rápido (G0), plano de trabalho XY (G17), ordem de cancelamento da compensação de diâmetro da ferramenta (G40), ordem de cancelamento da utilização de ciclos embebidos (G80), e escolha do modo de programação com coordenadas no modo absoluto (G90);
- Linha N104: troca de ferramenta (M06) – fresa de topo Ø40;
- Linha N410: troca de ferramenta (M06) – fresa cauda-de-andorinha Ø25;
- Linha N458: fim do programa (M30)

Estudado este exemplo – Modelo 1 – passe-se ao próximo: o Modelo 2.

4.2 – Modelo 2 – Operações 3D com superfícies complexas

O intuito deste modelo (Figura 52) é o de avaliar duas operações de desbaste e acabamento para superfícies mais complexas. Para estas entidades geométricas, será possível observar na simulação dos resultados, a influência das escolhas dos parâmetros das operações.

De uma forma geral, as operações 3D como o seu nome sugere, são operações nas quais o avanço em corte é feito por interpolação com utilização simultânea dos 3 eixos da máquina. Estas operações, sendo trabalhoso programá-las manualmente, beneficiam com a geração de percursos por computador, e posterior tradução em código de CN.

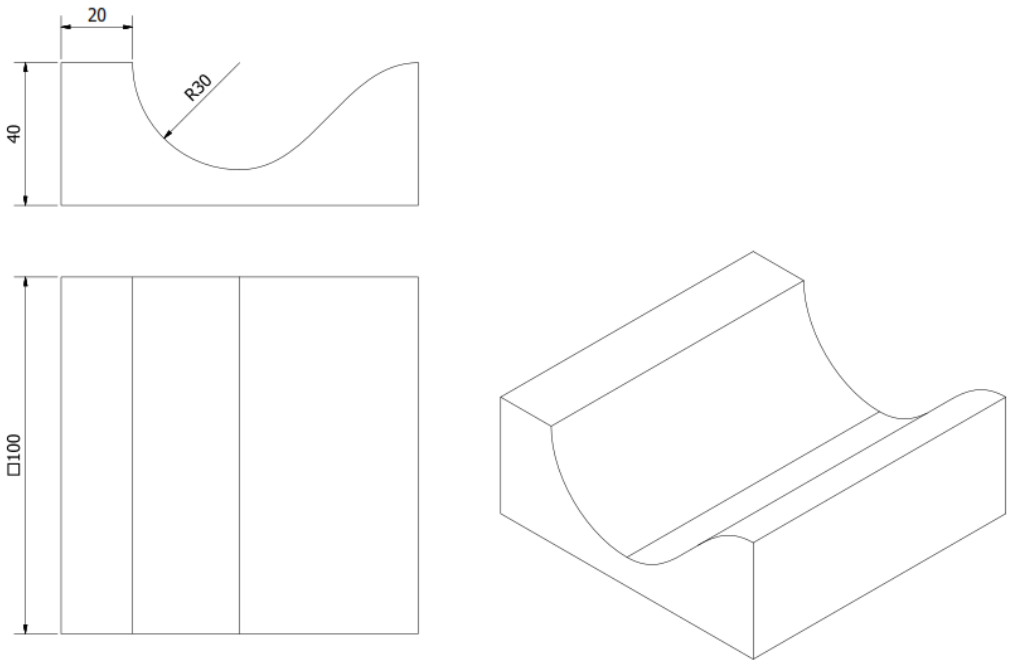


Figura 52 - Modelo 2.

O Modelo 2 é uma peça simples, pensada para ser fabricada em duas operações: uma de desbaste, e outra de acabamento específica para formas como a apresentada – designada de *flowline*[36]. A operação de desbaste usada neste exemplo aplica-se a superfícies 3D. No entanto o percurso gerado para essa operação é dividido em vários movimentos no plano XY, fazendo com que neste exemplo pudesse ser considerado uma sequência de percursos 2D.

Tendo em conta as 6 etapas consideradas na análise do Modelo 1, este exemplo considera apenas a configuração e simulação das operações de maquinagem.

[36] *Flowline*, do inglês "linhas de fluxo" é um termo usado para designar um conjunto de linhas que define um hipotético fluxo ao longo de uma superfície.

Etapa 4&5 – Seleção e configuração das operações/Revisão das operações

A primeira operação selecionada para este modelo é um desbaste 3D do tipo paralelo. Este desbaste divide o volume a remover em linhas paralelas entre si, de acordo com uma orientação específica, e repetida ao longo de vários planos de corte XY (Figura 53). Neste caso, escolheu-se orientar as linhas no seguimento da cavidade da peça.

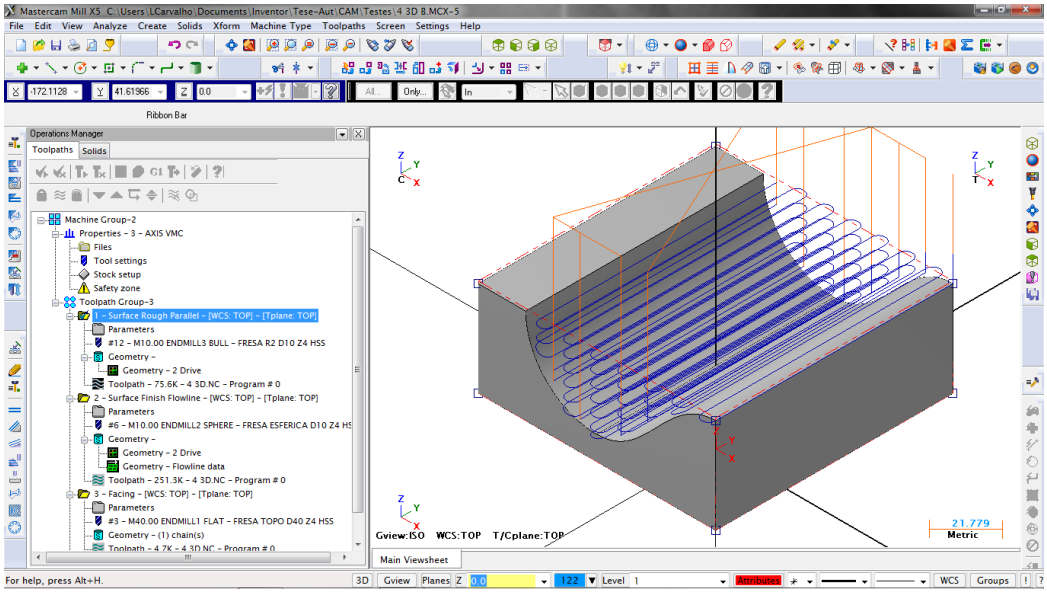


Figura 53 - Percurso de desbaste; versão inicial.

Para visualização da qualidade do desbaste, simula-se a remoção de material da operação, mostrando o resultado na Figura 54.

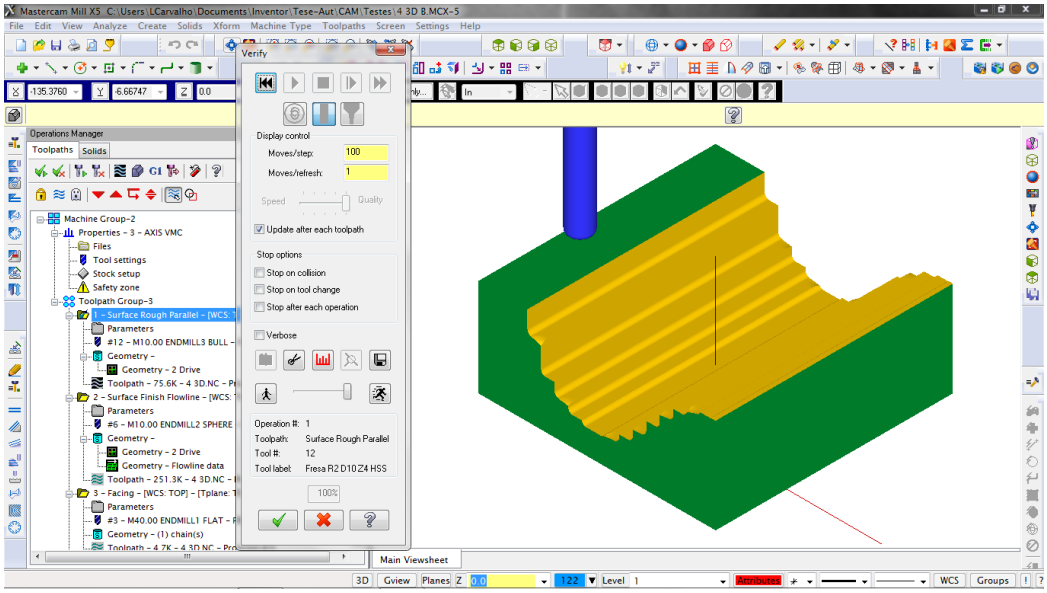


Figura 54 - Simulação de remoção de material.

Nesta fase é conveniente estipular um degrau vertical máximo (no direção do eixo Z), para que não reste demasiado material. Na realidade, este excesso de material pode ser excessivo, e ir além da capacidade de corte das ferramentas para acabamento.

Reduzindo-se a distância entre as linhas no plano XY de 6 para 3 mm, e entre planos de corte XY também de 6 para 3 mm, é possível melhorar a qualidade do desbaste inicial. Ilustra-se, na Figura 55, o percurso já com essa alteração.

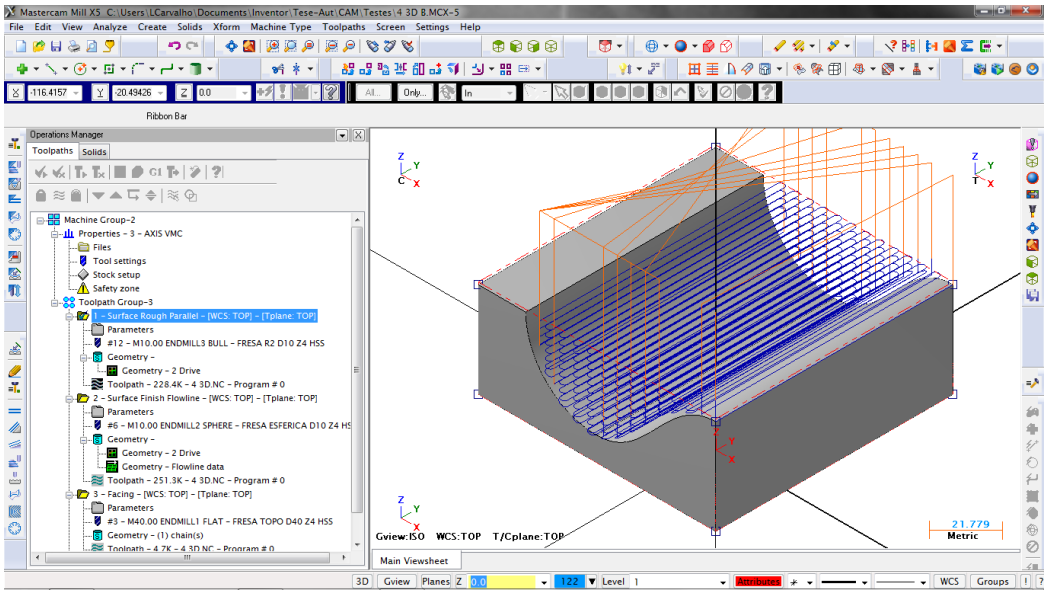


Figura 55 - Percurso de desbaste; versão refinada.

Como se pode observar, na figura, deu-se um aumento significativo no número de passagens. Na Figura 56 apresentam-se os resultados da simulação de remoção de material. Com restos de material em menor quantidade, este desbaste é naturalmente mais orientado para o resultado final.

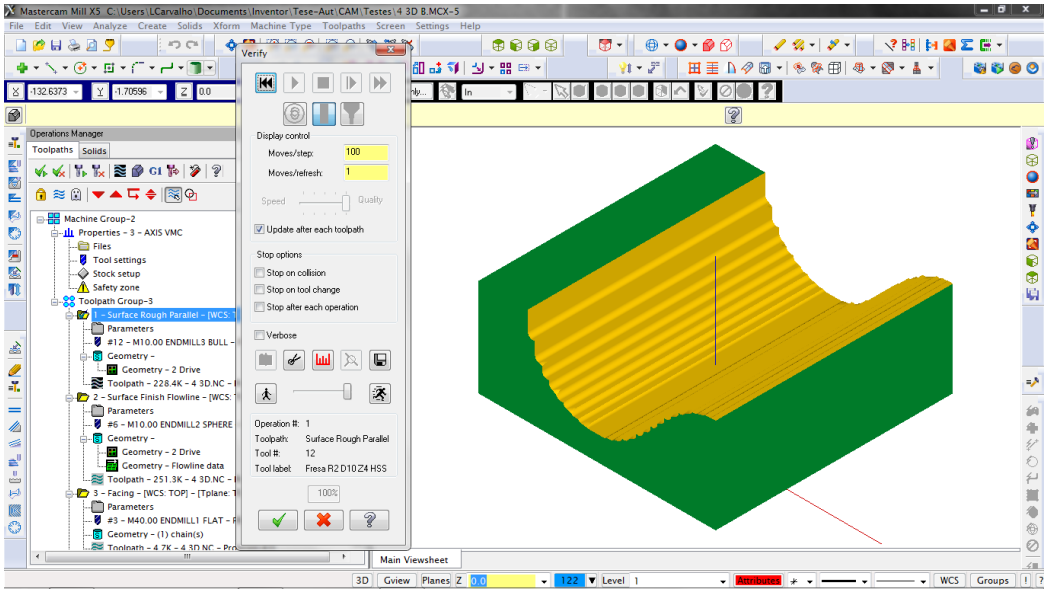


Figura 56 - Simulação de remoção de material; versão refinada.

Usando a mesma velocidade de corte, e um número maior de passagens corresponde a um aumento do tempo de ciclo. Nas Figura 57 eFigura 58 comparam-se os tempos de ciclos das duas operações.

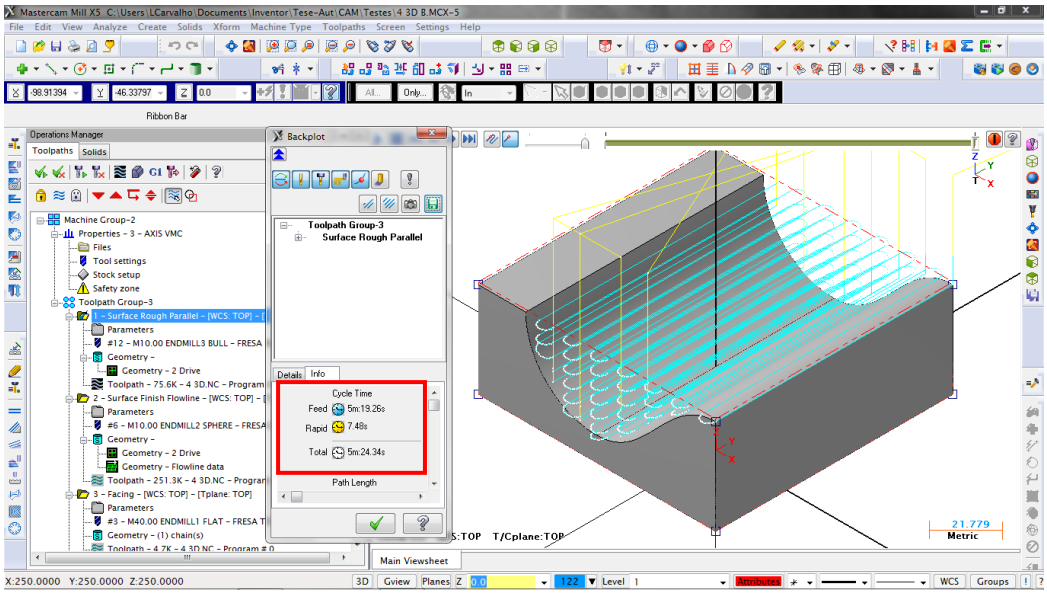


Figura 57 - Tempo de ciclo do percurso de desbaste; versão normal.

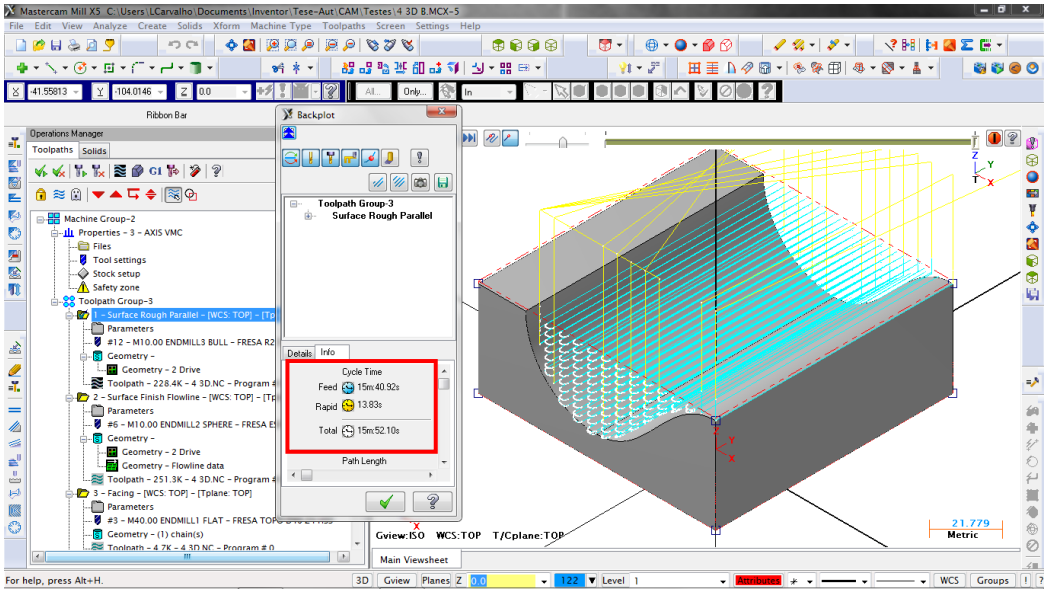


Figura 58 - Tempo de ciclo do percurso de desbaste; versão refinada.

Por comparação direta, observa-se que na Figura 57 o tempo de ciclo é de 5 minutos e 24 segundos, enquanto que na Figura 58 o tempo de ciclo apresentado é de 15 minutos e 52 segundos, ou seja, aproximadamente o triplo (cerca de 294%). Este tipo de resultado é útil, no sentido em que ajuda a pesar as consequências da procura de melhor qualidade, que por vezes pode levar a decisões erradas. Por exemplo, neste caso, essa procura pode não justificar o acrescento de tempo.

Adicionando uma operação de acabamento, ganha-se uma noção do resultado final, e dos fatores que o influenciam. Para este modelo usa-se uma estratégia orientada de acordo com a superfície complexa do modelo. A estratégia de acabamento em 3D do tipo *flowline* permite distribuir, homogeneamente, um percurso de maquinagem ao longo dessa superfície (Figura 59).

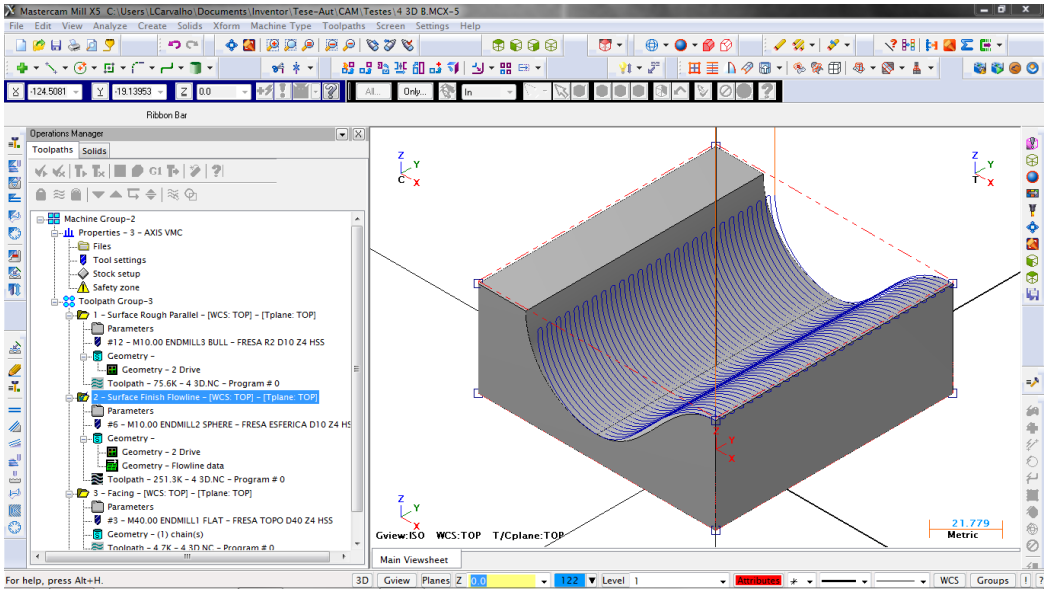


Figura 59 - Apresentação da operação de acabamento: *flowline*.

A orientação escolhida para esta geometria é a apresentada na Figura 59. Isto pode-se justificar pelo seguinte motivo: a distribuição homogénea obrigaria a que, em determinadas zonas do modelo – mais inclinadas – o percurso não fosse igualmente distribuído na direção do eixo Z. No que diz respeito ao acrescento de tempo de ciclo que esta operação provoca, esta é estimada em apenas 4 minutos e 30 segundos (por dedução ao tempo total do ciclo), como se mostra na Figura 60.

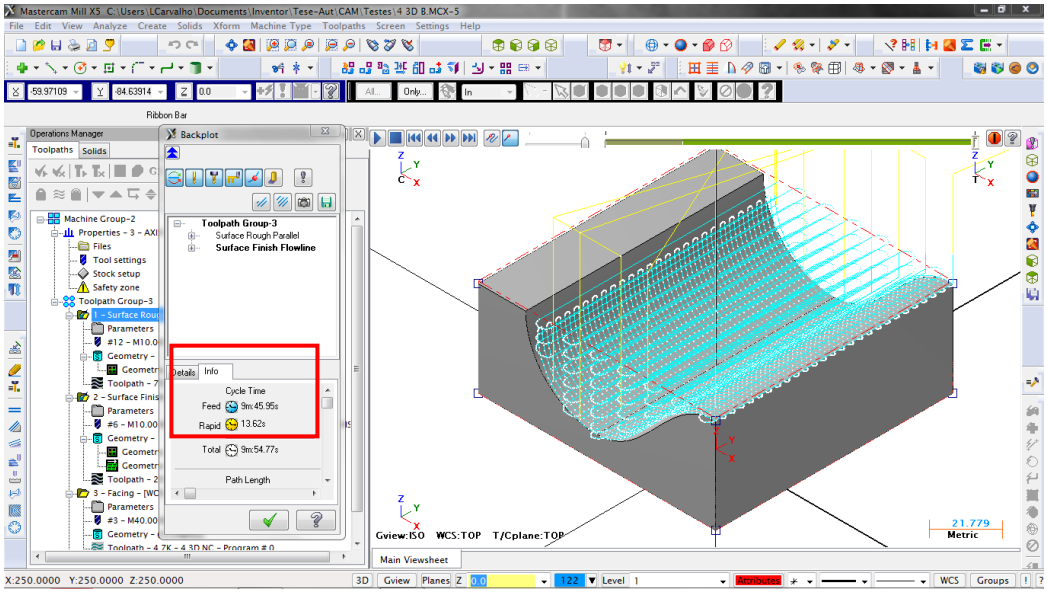


Figura 60 - Tempo de ciclo da operação de desbaste normal e acabamento normal.

Quanto à utilização de um desbaste com mais passagens juntamente com esta operação de acabamento, e verificação do resultado final, as Figura 61 eFigura 62 mostram que não há grandes benefícios em implementar este tipo de desbaste mais fino.

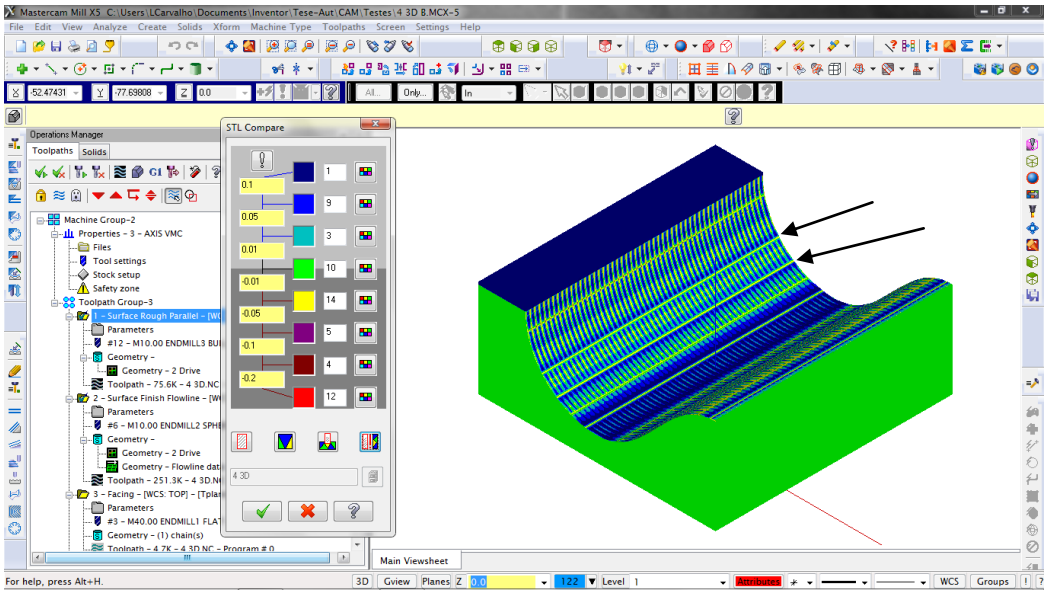


Figura 61 – Simulação da qualidade da operação de desbaste normal e acabamento normal.

Entre as Figura 61 eFigura 62, apenas se nota alguma diferença relativa às alturas dos "patamares" de desbaste (evidenciadas nas figuras). Desta forma, baseado na simulação, observa-se que a qualidade final é ditada pela operação de acabamento e não através da operação de desbaste. Estas são responsáveis pelo tempo de ciclo.

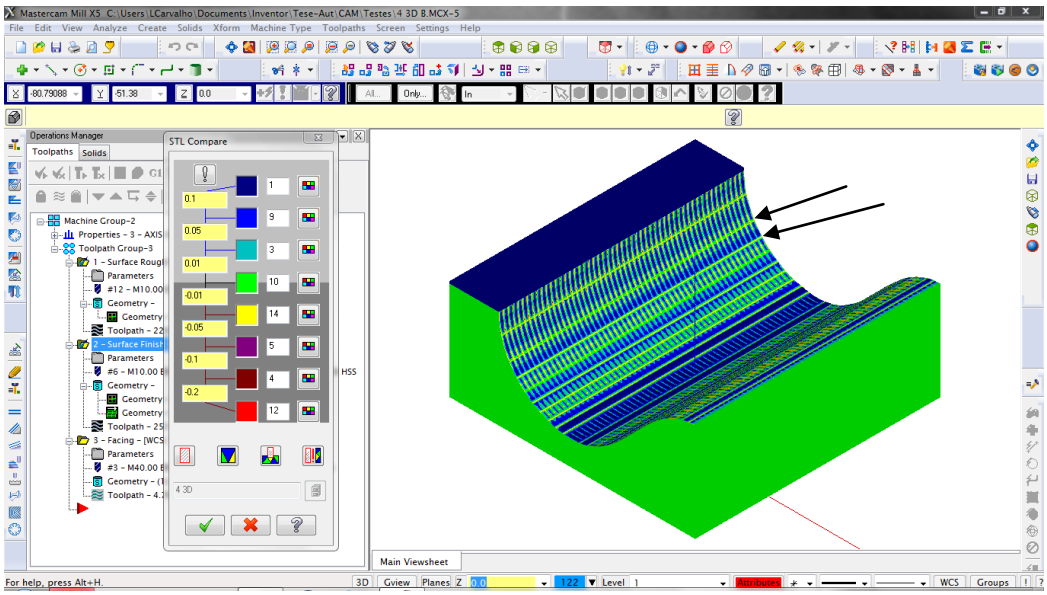


Figura 62 - Simulação da qualidade da operação de desbaste refinado e acabamento normal.

Uma questão que pode surgir é: se existe um aumento significativo do tempo de desbaste com o aumento da sua densidade, que tipo de aumentos são expectáveis com o refinamento das operações de acabamento?

Para responder a esta questão, reduz-se a altura expectável de resíduo de material de 0,1 mm para 0,05 mm[37]. Observe-se, na Figura 63, a estimativa do tempo de ciclo.

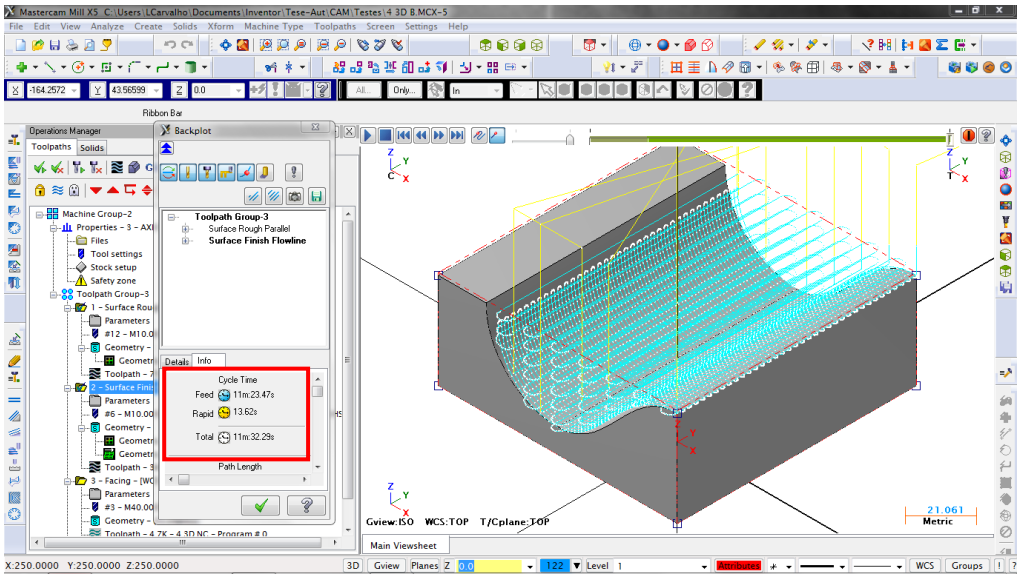


Figura 63 - Tempo de ciclo da operação de desbaste normal e acabamento refinado.

Constate-se que do tempo de ciclo anterior (Figura 60), de 9 minutos e 54 segundos, este caso difere em 1 minuto e 38 segundos, representando um aumento de cerca de 16%. (no resultado anterior, deu-se um aumento de 194%).

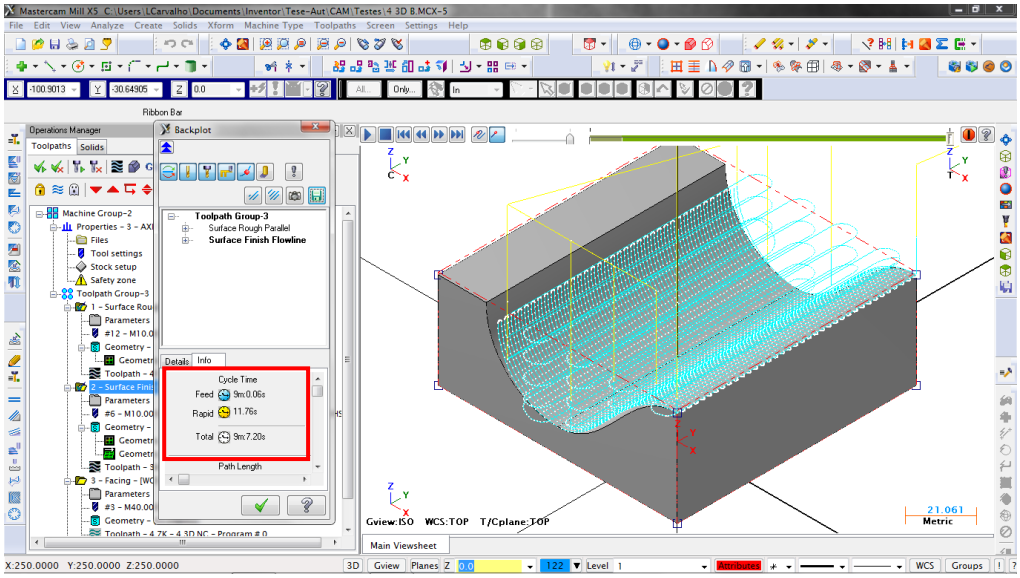


Figura 64 - Tempo de ciclo da operação de desbaste grosseiro e acabamento refinado.

Na Figura 64 mostra-se o percurso alterado, com uma operação de desbaste grosseiro, mantendo o acabamento refinado, poupando, no total, 2 minutos e 25 segundos, tornando esta a solução mais rápida.

[37] Na operação *flowline*, o parâmetro-chave que define a sua densidade é o *scallop height*. Este termo, *scallop*, representa o resquício de material que reside entre duas passagens consequentes de uma ferramenta de corte.

Relativamente à qualidade das soluções até aqui planeadas, comparem-se as Figura 65 e Figura 66.

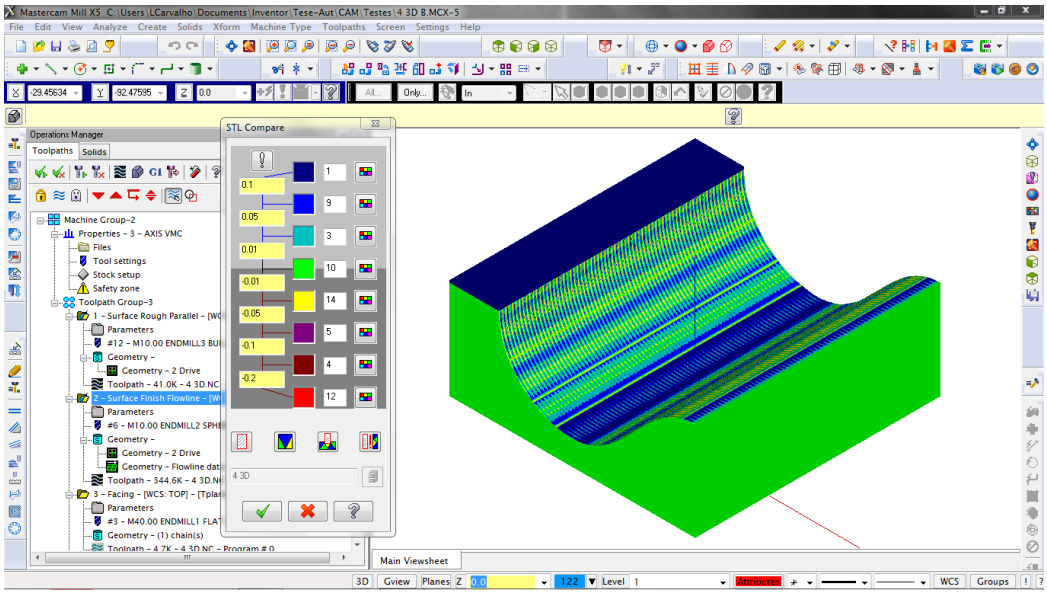


Figura 65 - Simulação da qualidade da operação com desbaste normal e acabamento refinado.

A Figura 65 corresponde à solução mais rápida, com desbaste de grosseiro, e acabamento refinado. Já na Figura 66, ilustra-se a solução de desbaste e acabamentos refinados. Aqui, mais uma vez, à exceção dos "patamares" de desbaste no eixo Z, é difícil distinguir a qualidade dos resultados. No entanto, estima-se que uma solução demore 9 minutos e 7 segundos, e que a outra demore 22 minutos.

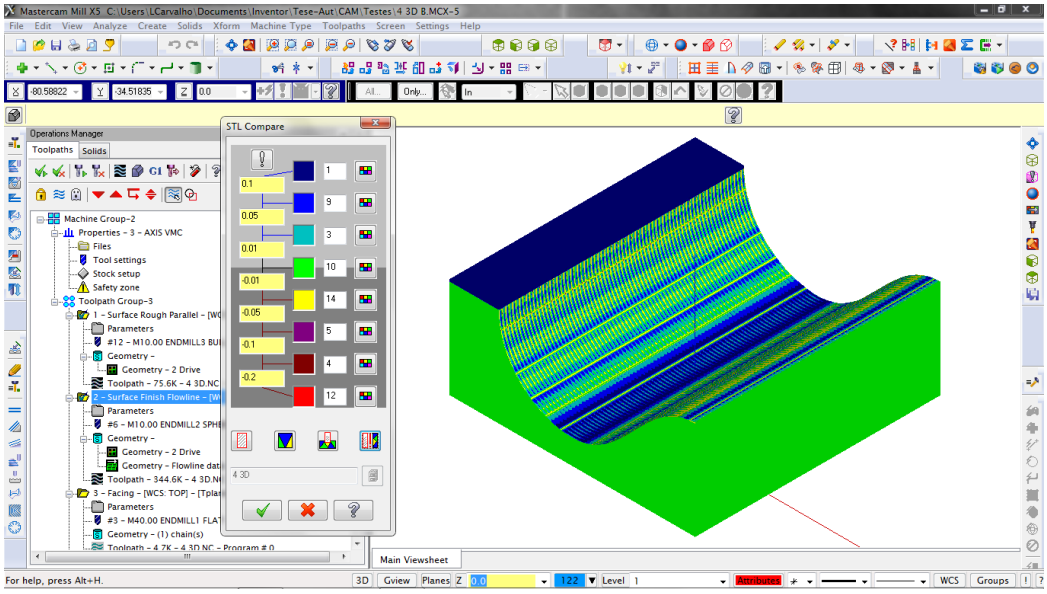


Figura 66 - Simulação da qualidade da operação com desbaste grosseiro e acabamento refinado.

O efeito contrário ao que foi observado, seria o da degradação da operação de acabamento. Para dar uma noção desse efeito, diminui-se o critério de qualidade da operação de acabamento, aumentando o valor do *scallop* para 0,2 mm. O resultado apresenta-se na Figura 67.

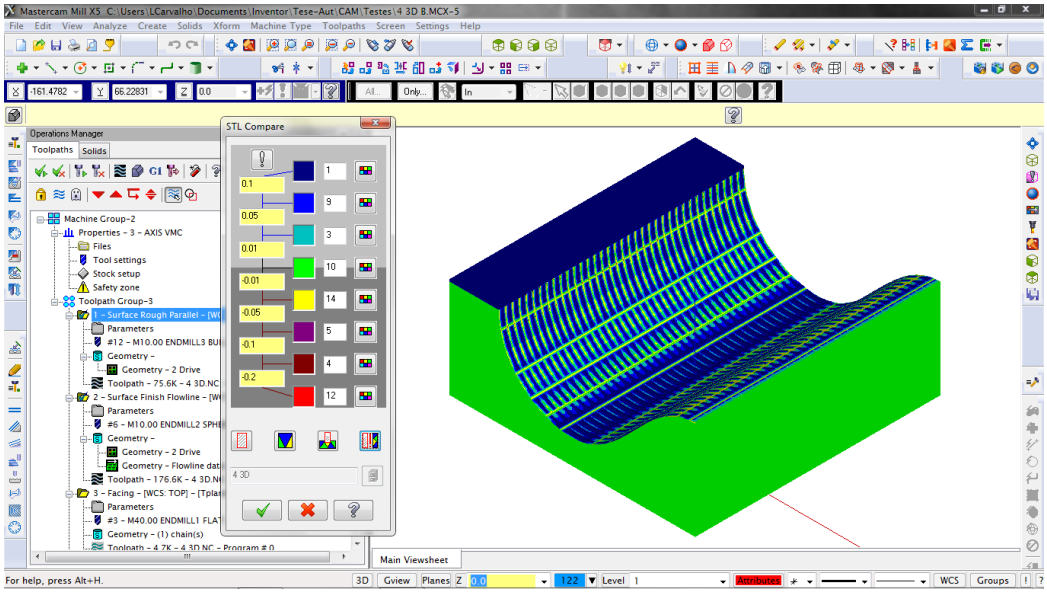


Figura 67 - Simulação da qualidade da operação com desbaste normal e acabamento grosseiro.

Como se pode observar, na Figura 67, o aumento do *scallop* diminui a qualidade do resultado final. Para terminar estes testes, altera-se a orientação da operação de desbaste, de modo a que fique paralela à operação de acabamento. Na Figura 68 mostra-se o percurso e o tempo de ciclo para a operação de desbaste.

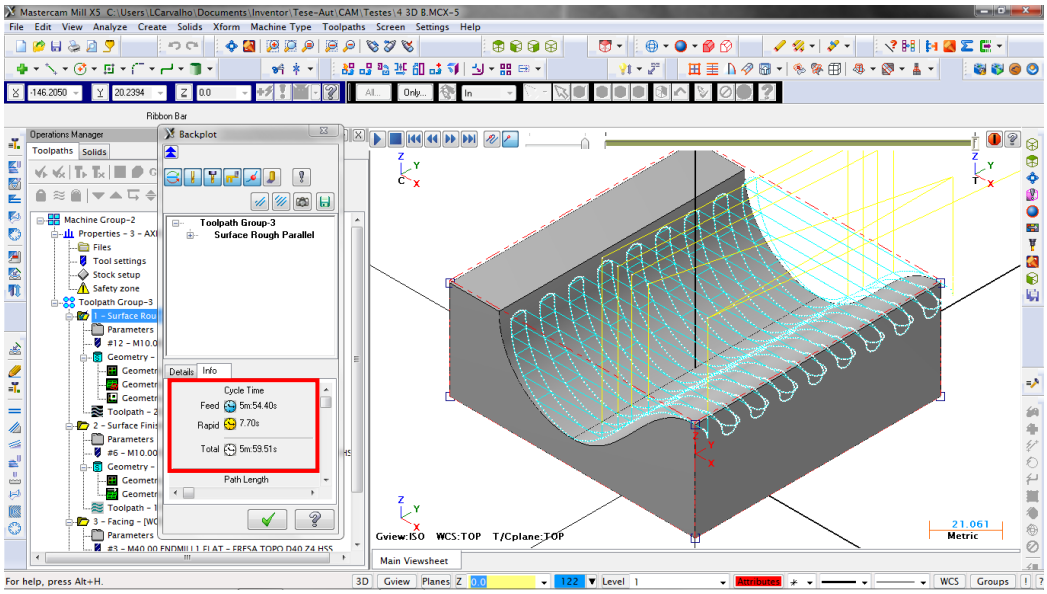


Figura 68 - Percurso de desbaste com rotação de 90°.

A operação de desbaste adicionou cerca de 25 segundos ao tempo de ciclo.

Relativamente à qualidade do acabamento final, com este tipo de desbaste, os resultados apresentam-se nas Figura 69 eFigura 70, onde se aplicam os dois acabamentos até aqui praticados.

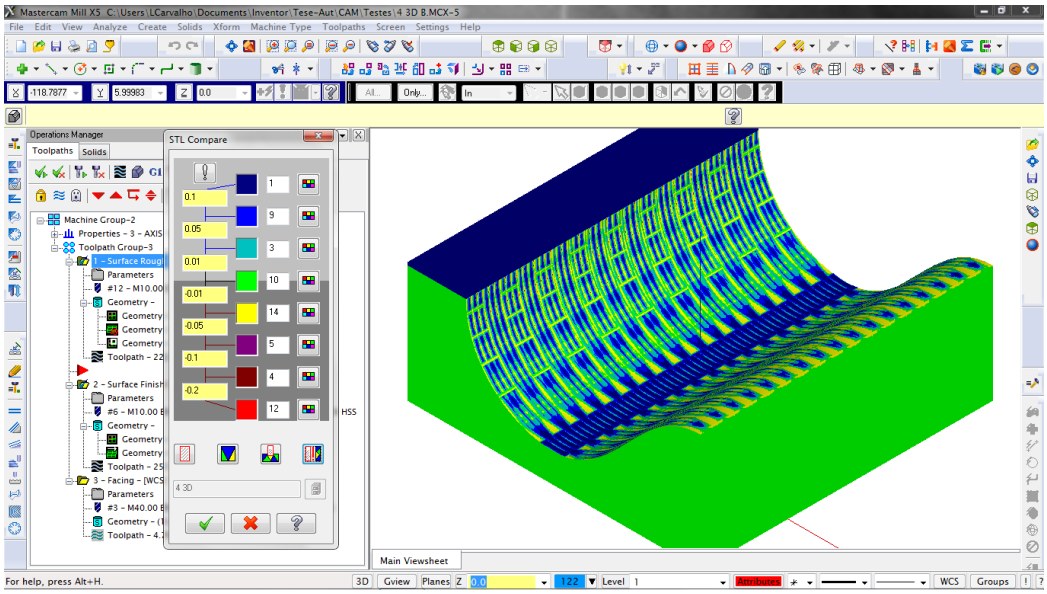


Figura 69 - Simulação da qualidade da operação com desbaste a 90° e acabamento normal.

Nestes dois casos, pode-se observar outro fator que influencia a qualidade. Nas Figura 69 eFigura 70 existe sobreposição da direção de corte de uma operação para a outra. Isto resulta na existência de zonas em que o acabamento trabalhará sem cortar material, e resulta também numa superfície menos uniforme (em simulação). Para contrariar isso, devem-se definir, nos parâmetros das operações, sobreespessuras devidamente calculadas, para serem trabalhados posteriormente no acabamento.

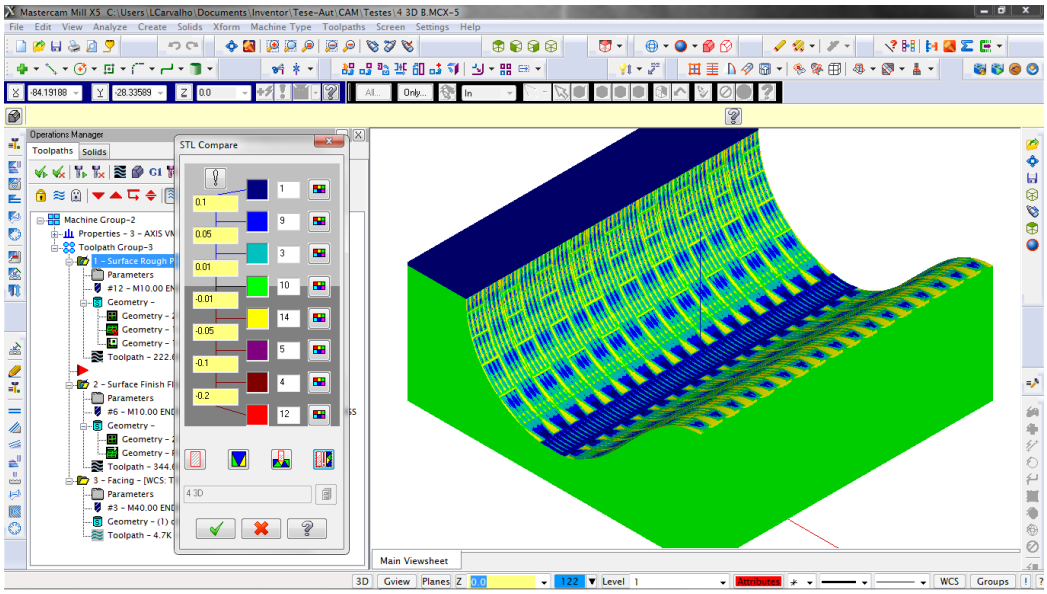


Figura 70 - Simulação da qualidade da operação com desbaste a 90° e acabamento refinado.

Terminado este exemplo faz-se um resumo do que se realizou com o Modelo 2:

1. Definiram-se duas estratégias para desbaste e acabamento;
2. Exploraram-se cenários de interação entre ambas;
3. Obtiveram-se estimativas de tempos de ciclo;
4. Comparam-se resultados de qualidade e uniformidade do acabamento;
5. Recolheram-se informações respeitantes à relação qualidade-tempo, e de que forma algumas operações influenciam essa relação.

Neste exemplo aprofundou-se a apresentação do CAM como apoio na tomada de decisão, mas mais particularmente no que diz respeito ao ganho de conhecimentos. Esses conhecimentos obtidos podem agora ser aplicados noutras situações e inclusive, serem validados ou não em ensaios reais.

Terminado o exemplo com o Modelo 2, reveja-se de seguida o Modelo 3.

4.3 – Modelo 3 – Operações 3D com orientação radial

Enquanto que o Modelo 1 exemplifica o básico da configuração de um modelo para produção, e o Modelo 2 aprofunda o apoio à tomada de decisão e planeamento tecnológico, o Modelo 3 tem o intuito de fornecer algumas considerações adicionais sobre a escolha das estratégias de maquinagem, para peças com orientação radial (Figura 71)

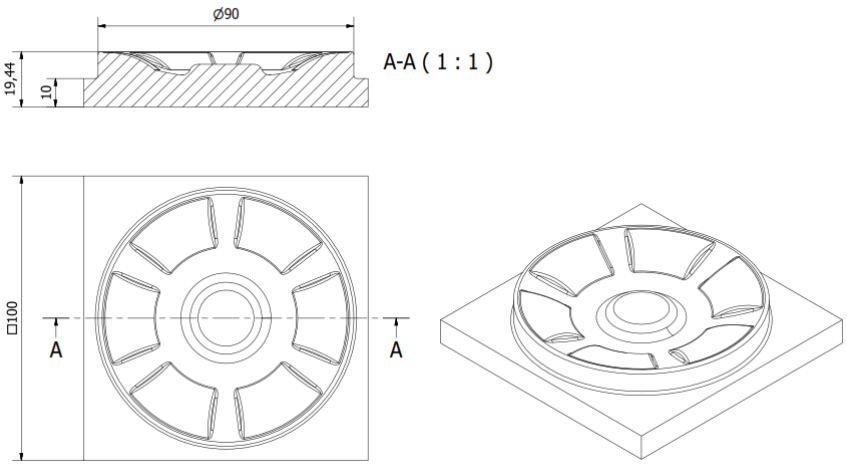


Figura 71 - Modelo 3.

O Modelo 3 consiste numa peça que emula, até certo ponto, uma forma radial genérica que pode ser associada, por exemplo, a uma jante de um automóvel.

As particularidades deste Modelo são as seguintes:

- 6 zonas periféricas – aqui designadas de “alvéolos”;
- Um maciço central, plano no seu centro.

À semelhança do Modelo 2, o exemplo deste modelo considerará apenas as etapas 4 e 5.

Etapas 4&5 – Seleção e configuração das operações/Revisão das operações

A forma apresentada na Figura 72 sugere a aplicação de estratégias de maquinagem radiais. Tal como foi aplicado no Modelo 2, também aqui se poderão procurar formas de cruzar as passagens de desbaste com as passagens de acabamento, de forma a melhorar a qualidade global da maquinagem.

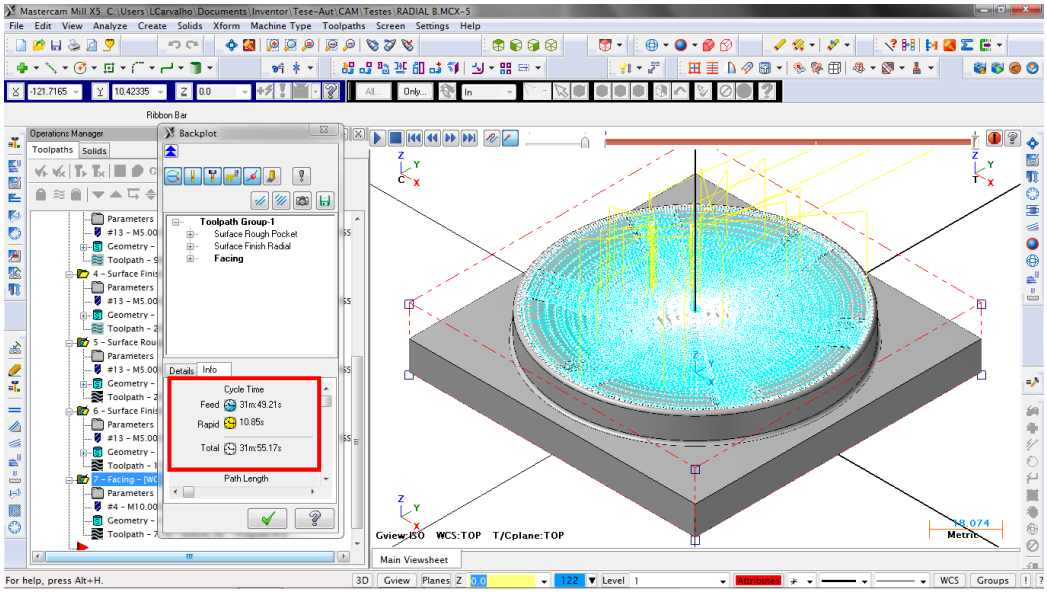


Figura 72 - Geração de percurso e visualização do tempo de ciclo - estratégia 1.

A Figura 72 mostra a definição de um percurso de maquinagem com duração de 31 minutos e 55 segundos, e que combina 3 estratégias de maquinagem: desbaste de cavidades em espiral; acabamento de superfícies com distribuição radial; facejamento final do maciço central. A Figura 73 apresenta uma estratégia alternativa, com recurso a uma ferramenta de maior diâmetro, tentando reduzir o tempo de ciclo.

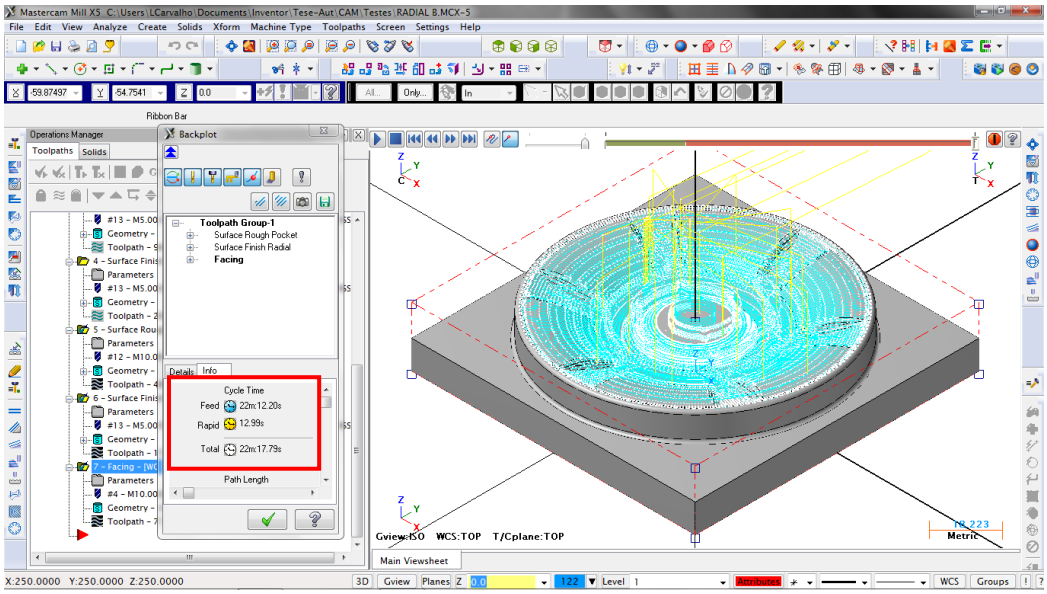


Figura 73 - Geração de percurso e visualização do tempo ciclo - estratégia 2.

Como se pode observar, o tempo de ciclo foi diminuído para 22 minutos e 17 segundos. A nível de resultados, apresentam-se as Figura 74 eFigura 75.

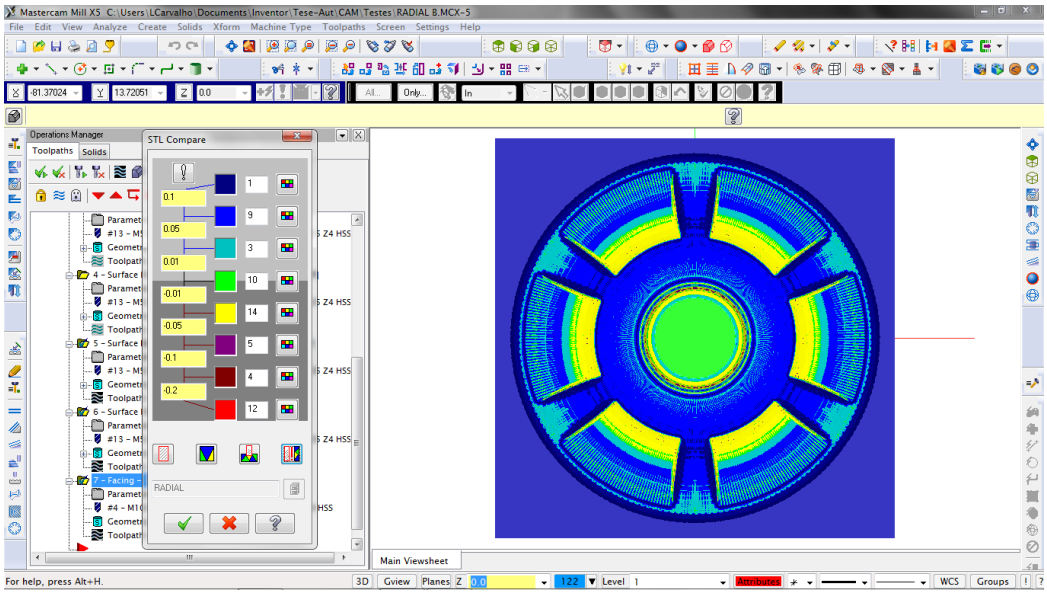


Figura 74 - Simulação da qualidade da operação - estratégia 1.

Na Figura 74 é utilizada a estratégia com o desbaste mais fino. Na Figura 75, o desbaste é mais grosseiro. Aqui a diferença reside essencialmente na uniformidade do resultado final. É de notar, na Figura 75, o maior desbaste entre os "alvéolos", comparativamente ao resultado da Figura 74.

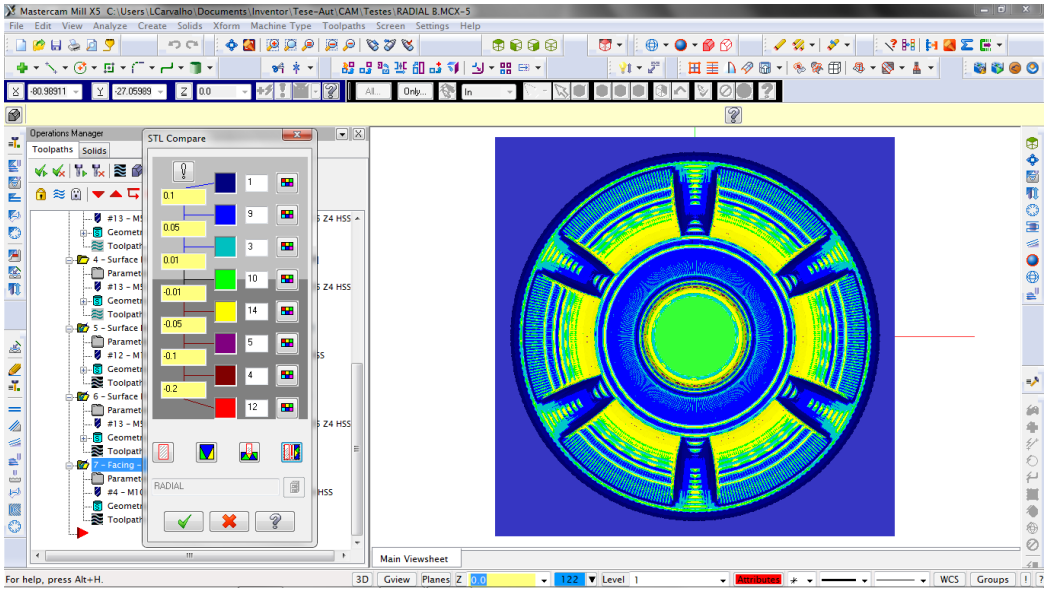


Figura 75 - Simulação da qualidade da operação - estratégia 2.

Relativamente à utilização de outras estratégias não tão adaptadas a formas com orientação radial, ilustra-se nas Figura 76 eFigura 77 o resultado do desbaste com uma estratégia paralela (usada no desbaste do Modelo 2), e o resultado final, já com as operações de acabamento.

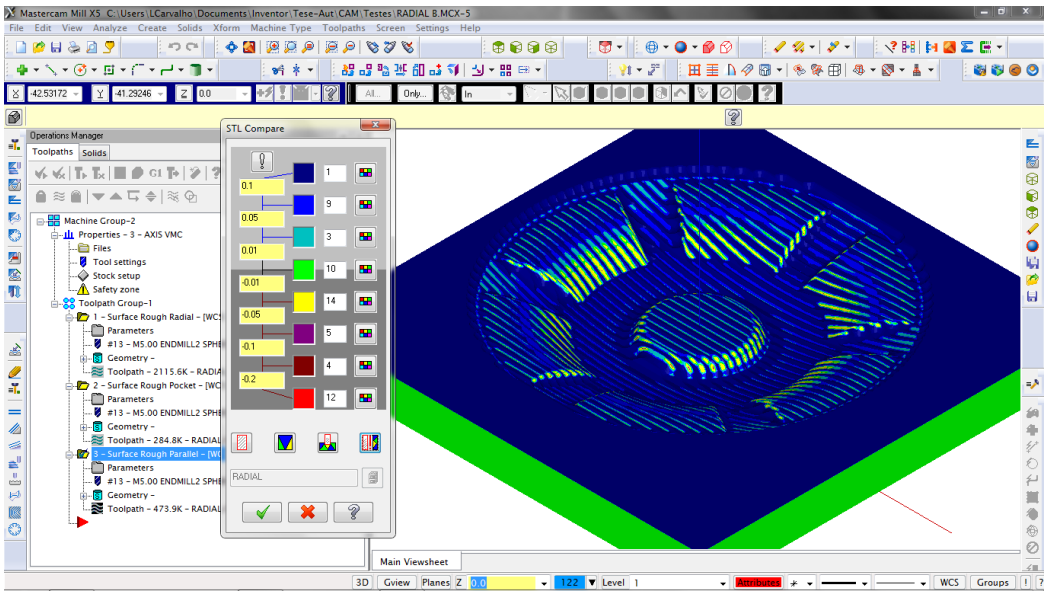


Figura 76 - Simulação da qualidade da operação de desbaste em modo paralelo.

É possível observar, nas Figura 76 eFigura 77, o impacto que uma distribuição não radial tem numa peça com essa orientação. A pouca uniformidade do desbaste poderá provocar fraco funcionamento da ferramenta de acabamento, e mesmo com a operação de acabamento, a peça poderá ainda ficar com marcas do desbaste anterior[38].

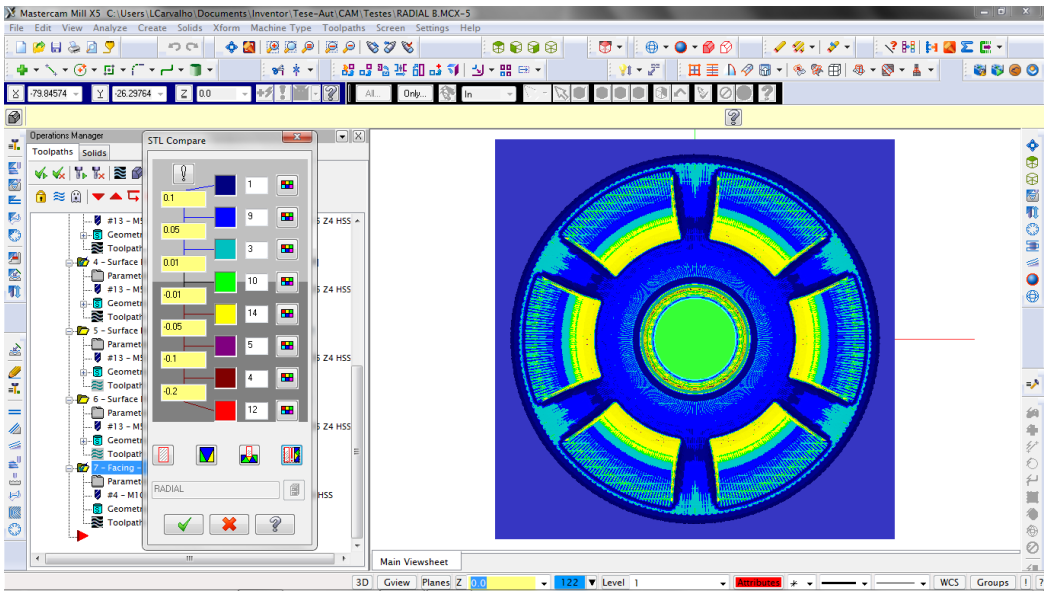


Figura 77 - Simulação da qualidade da operação de desbaste paralelo com acabamento.

No próximo capítulo será iniciada a configuração da máquina virtual. Esta fase englobará uma breve explicação do seu processo, particularizando a estrutura e funcionamento do pós-processamento. Em linha com essa configuração será feito um estudo compreensivo das capacidades de uma máquina em particular.

[38] Aqui apenas se pode assumir a hipótese de os resultados simulados serem semelhantes à realidade. Para os devidos efeitos deste estudo, considera-se que, com base nas simulações, há de facto uma base que permite extrapolar para uma operação real.

5 – Configuração do Mastercam para a máquina Okuma MC-40VA

Os exemplos de utilização do programa, nas diversas fases de planeamento das operações de maquinagem, na geração do código de CN, foram úteis para mostrar o modo de funcionamento do *software*. Em linha do planeamento da produção, o passo seguinte consiste na configuração desse *software* para uma máquina em particular.

No âmbito desta dissertação, essa configuração será aplicada ao programa em questão – Mastercam X5 – e indo de encontro às características da máquina Okuma MC-40VA, e do seu controlador OSP5020M, culminará na criação de uma máquina virtual e respetivo pós-processador.

Este capítulo trata dessa configuração, começando por expor as fases em que consiste, com base na estrutura apresentada no capítulo 3. Faz-se de seguida uma explicação do conceito de pós-processamento, e dos recursos que ele envolve. Posteriormente serão apresentadas as características da máquina e do seu controlador, no contexto da configuração da máquina virtual e do seu pós-processador. Por fim expõe-se a edição do pós-processador.

5.1 – Criação da máquina virtual

A máquina virtual, conforme explicado no capítulo 3 (pp. 34), constitui uma base de dados na qual estão contidas informações usadas na simulação das operações de maquinagem e na geração de código de CN.

Estas informações compreendem:

- Parâmetros gerais de funcionamento da máquina;
- Parâmetros específicos da organização e funcionamento de componentes;
- Parâmetros associados ao ficheiro de controlo (incluindo pós-processador);
- Definições de bases-de-dados associadas;
- Outras opções.

O primeiro passo da configuração da máquina virtual é o de pesquisar informação. Como se mostrará adiante, é necessário haver uma recolha sensível dos seguintes dados:

A. Informações relacionadas com o tipo de máquina – este grupo é o mais elementar, mas também imprescindível. Aqui incluem-se por exemplo o número de eixos da máquina, a sua interação com a mesa de trabalho, o número de *spindles* e a organização destes elementos;
B. Capacidades de trabalho da máquina – este conjunto de informações é importante na medida em que limita a utilização do *software* de acordo com as possibilidades da máquina (i.e. dimensões da mesa de trabalho, velocidades máximas dos eixos, entre outros);

C. Dimensões e formas da máquina e acessórios – a utilização desta informação será apenas na componente de simulação e deteção de colisões. A sua importância está diretamente relacionada com a utilização destas funcionalidades de *software*;

D. Existência de funções auxiliares da máquina – aqui as informações dizem respeito à utilização de sistemas de refrigeração, controlo de abertura de guardas de proteção, sopragem de ar, entre outros. É crucial conhecer bem estas particularidades, pois de acordo com as preferências do utilizador, poderão ser implementadas no pós-processamento, para que possam ser incluídas no código de CN;

E. Particularidades do controlador da máquina – este grupo de informação é o mais relevante, pois é o que, quando bem conhecido e compreendido, permite programar um pós-processador com o maior grau de confiança possível, e implementar no mesmo outras funções que ajudem a automatizar todo o sistema. Isto é útil, por exemplo, no caso da máquina possuir sistemas de medição de *offset* de ferramenta, ou sistemas de controlo de força ou controlo de desgaste da ferramenta;

F. Preferências da oficina/utilizador – aqui incluem-se opções como por exemplo a escrita de informação do programa ou das ferramentas no programa de código de CN. Assume-se que determinadas opções estão associadas à peça em si (ao trabalho em particular), e como tal não constam como preferências da máquina.

O passo seguinte está relacionado com a aplicação das informações recolhidas, através da utilização da interface[39] do Mastercam. Os pontos desta fase são:

1. Criação da árvore[40] da máquina e parametrização dos seus componentes – este ponto diz respeito às velocidades de avanço (rápido e em corte) máximas e mínimas para cada eixo, assim como os seus limites e posição neutra[41];
2. Criação do modelo 3D da máquina – embora não se considere um ponto crucial da definição da máquina virtual, a sua realização acrescenta valor à configuração da máquina virtual, permitindo-lhe simular os percursos gerados;
3. Definição das configurações gerais da máquina – este ponto é relativo por exemplo à posição de troca de ferramenta, velocidades máximas e mínimas do *spindle*, e existência de refrigeração[42];
4. Definição dos parâmetros do ficheiro de controlo – aqui a tarefa torna-se mais complexa. Observando mais uma vez a Figura 13 (pp. 34), no capítulo 3, percebe-se que este ficheiro contém parâmetros que são aplicados a qualquer máquina virtual, e outros que dependem do pós-processador, e que são aplicados

[39] Em oposição com a modificação do ficheiro do pós-processador, que é feita de forma direta.

[40] Não confundir com o termo árvore, usado para designar o local onde se encontra o *spindle*. Aqui este termo é aplicado para designar uma estrutura organizacional dos vários componentes que compõe a máquina ferramenta.

[41] Aqui prefere-se usar este termo, ao invés do termo zero, pois poderá criar ambiguidade. Zero da máquina pode não ser necessariamente a posição neutra da mesma.

[42] Embora neste ponto se controle a existência de refrigeração, será ainda necessário definir os códigos que traduzem esta função.

por via deste. Isto é, alguns parâmetros necessitam de suporte por parte do pós-processador. Posto isto refira-se que aqui se inserem tanto definições de funções auxiliares da máquina, como particularidades do controlador, e preferências do utilizador;

Por fim o último passo é o da aplicação das informações recolhidas, agora de forma direta: através da edição do pós-processador: neste ponto incluem-se virtualmente todos os aspetos antes listados (de A a F). Como se irá abordar no próximo subsecção, o funcionamento do pós-processador é favorável à integração de todos estes dados.

Apresenta-se, na Figura 78, um diagrama que os relaciona:

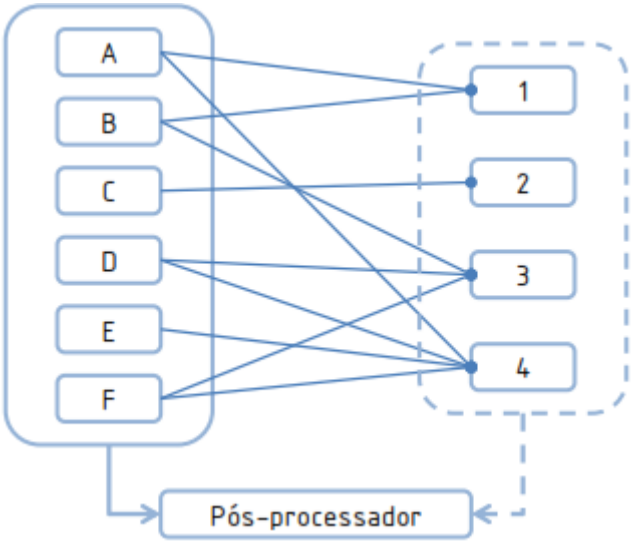


Figura 78 - Relações entre fontes de recolha de dados e aplicações dos mesmos.

5.2 – Configuração de um pós-processador

Pós-processamento é a fase imediatamente anterior à obtenção de código de comando numérico. Consiste num processo de tradução, automatizado, que – tal como mencionado nos capítulos anteriores – converte o caminho planeado na interface do *software*, entretanto já transformado no código de formato APT (ficheiro *.NCI). Na Figura 79 explica-se este funcionamento.

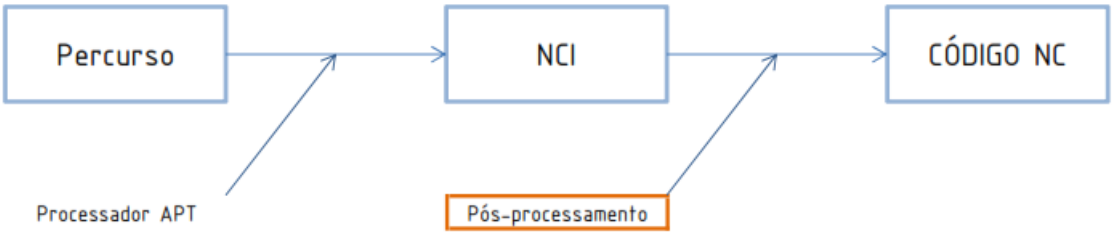


Figura 79 - Pós-processamento.

O pós-processador reúne, essencialmente, dois aspetos associados ao processo de maquinagem:

1. Opções e capacidades da máquina e do seu controlador, configuráveis na interface, e que são indiretamente definidas nas rotinas de pós processamento. Inserem-se neste campo, por exemplo, as velocidades máximas do *spindle*, a resolução do controlador, e as opções escolhidas no ficheiro de controlo, entre outros;
2. Preferências do utilizador, relativas ao centro de maquinagem e interpretação dos percursos e NCI gerados. Aqui figuram opções como, por exemplo, a inserção de comentários relativos às designações das ferramentas, utilização de refrigeração, ou escolha de sistemas de eixos. Incluem-se ainda neste ponto outras opções e configurações inseridas e customizadas pelo utilizador. Referem-se por exemplo códigos modais e auxiliares específicos, rotinas de verificação de erro, acesso especial a informações de componentes, ou rotinas de paragem para verificação da qualidade de determinadas operações.

O pós-processador está sempre associado a uma máquina virtual, representativa de uma máquina real. No entanto essa associação não é exclusiva. Cabe ao utilizador exercer a sua preferência. Tipicamente será uma das duas:

- Um pós-processador para cada máquina virtual;
- Dois ou mais pós-processadores para cada máquina virtual, em que cada um cobre utilizações distintas das suas capacidades.

No sentido contrário, um único pós-processador pode estar associado a mais do que uma máquina, caso estas tenham modos de funcionamento idênticos.

5.2.1 – Estrutura de um pós-processador no Mastercam X5

No Mastercam X5 o pós-processamento dá-se com recurso a um programa descrito num ficheiro de texto. Esse ficheiro de texto deverá ter a extensão *.pst, para que o *software* o reconheça como um de pós-processador. O processo em si é realizado por uma aplicação incorporada no Mastercam X5, designada de MP[43]. Esta aplicação pode considerada uma DAPP – *Design Aid for Post-Processing* – um conceito desenvolvido pela IBM nas décadas de 1950 e 1960 [1]. Esta aplicação auxilia o utilizador na configuração de pós-processadores, mantendo o seu "núcleo" (comum a todos na leitura de CLDATA), sem necessidade de edição. Este MP.dll é então responsável por interpretar os comandos e funções inseridos no ficheiro do pós-processador[44]. A linguagem de programação usada no pós-processador é específica ao MP.dll e ao Mastercam X5 – embora haja um paralelismo entre alguns comandos usados aqui e em outras linguagens mais comuns, como C++.

O Mastercam X5 fornece alguns pós-processadores, prontos a usar, e com a capacidade de gerar código de comando numérico que terá de ser sujeito a minuciosas revisões:

[43] Apenas se pode especular quanto ao significado da sigla "MP". O ficheiro ao qual a aplicação recorre é conhecido como MP.dll.

[44] Para desambiguação e melhor compreensão por parte do leitor, utilizarei o termo "pós-processador" para designar o ficheiro de texto editável (.pst), e "aplicação de pós-processamento" para designar o ficheiro MP.dll.

- Genéricos (como o que foi usado nos exemplos do capítulo anterior);
- Típicos de algumas marcas de controladores (por exemplo: Fanuc, Haas);

Estes pós-processadores são fornecidos com o intuito de serem modificados pelo utilizador, e adaptados ao controlador a usar no centro de maquinagem. Apesar de ser possível obter-se um pós-processador específico para um controlador em particular, a configuração e edição de pós-processadores é uma proficiência altamente útil, que é possível aprender, e cujos resultados são facilmente apreciáveis, pois o código de comando numérico gerado através de um pós-processador robusto requer reduzidas ou mesmo nulas alterações. Na Tabela 1 listam-se os constituintes de programação de um pós-processador básico. Embora a sua sequência[45] não seja relevante (já que a leitura deste ficheiro é realizada previamente pelo MP.dll), recomenda-se que seja respeitada, já que, além de ser praticada em todos os pós-processadores do Mastercam – e nesta fase é bom haver consistência – retêm uma lógica que depois de assimilada, será apreciada.

- Leitura de parâmetros-operação e parâmetros-configuração do ficheiro NCI[46];
- Leitura de definições da máquina virtual e da configuração do controlo;
- Articulação das opções registadas com os comandos definidos no NCI;
- Escrita sequencial dos blocos de comando com as informações processadas.

[45] Apesar de ter sido feito um esforço para ilustrar a sequência de funcionamento do pós-processador, e assim preservar a sua forma, com vista a valorizar a organização de conteúdos da tabela, alguns blocos não se encontram organizados como originalmente estão no ficheiro .pst.

[46] Estes conceitos serão melhor explicados nas páginas seguintes.

Tabela 1 - Estrutura e composição típicas do pós-processador básico no Mastercam.

INICIALIZAÇÃO DO PÓS-PROCESSADOR	COMENTÁRIOS INICIAIS	VERSÃO; MELHORAMENTOS; MÁQUINA; CONTROLADOR
	INICIALIZAÇÃO DE SWITCHES	DEFINIÇÕES DE OPÇÕES - LIGAÇÃO AO *SOFTWARE*
	INICIALIZAÇÃO DE VARIÁVEIS	DEFINIÇÃO DE VALORES - LIGAÇÃO AO *SOFTWARE*
	DEFINIÇÃO DE *STRINGS*	CUSTOMIZAÇÃO DE TEXTO DE CÓDIGOS DE CN E MENSAGENS
	DEFINIÇÃO DE FORMATOS	CUSTOMIZAÇÃO DE FORMATAÇÕES DE NÚMEROS
	AFECTAÇÃO DE FORMATOS A VARIÁVEIS	APLICAÇÃO DE FORMATOS A VARIÁVEIS
ROTINAS GERAIS	BLOCOS DE COMEÇO DE FICHEIRO	INSERÇÃO DE INFORMAÇÕES: MÁQUINA, FERRAMENTAS, MATERIAIS, DATA, HORA
	BLOCOS DE PÓS-PROCESSAMENTO	PROCESSAMENTO DE LINHAS DO CÓDIGO NCI: IDENTIFICAÇÃO DE OPERAÇÕES E CÓDIGOS
	BLOCOS AUXILIARES	BLOCOS QUE TRADUZEM OPÇÕES EM ALTERAÇÕES DE OUTROS BLOCOS
	BLOCO DE FIM DE FICHEIRO	BLOCO QUE ENCERRA O FICHEIRO DE CÓDIGO COMANDO NUMÉRICO
ROTINAS DE IN/OUT	BLOCOS DE INPUT	BLOCOS QUE LÊEM DEFINIÇÕES DE CONTROLO, DA MÁQUINA VIRTUAL OU FICHEIRO NCI
	BLOCOS DE OUTPUT	BLOCOS QUE AGREGAM RESULTADOS DE CÁLCULOS E TRADUÇÕES DE PÓS-PROCESSAMENTO E ESCREVEM NO FICHEIRO NC
	BLOCOS DE PRÉ-PÓS-PROCESSAMENTO	BLOCOS QUE REALIZAM OPERAÇÕES INICIAIS, ANTES DE HAVER LEITURA DO FICHEIRO NCI
	BLOCOS DE LEITURA ESPONTÂNEA	BLOCOS QUE REALIZAM A LEITURA (IDÊNTICOS A BLOCOS DE INPUT), MAS DE UTILIZAÇÃO CUSTOMIZÁVEL
ROTINAS DE CONFIGURAÇÃO DE *SOFTWARE*	BLOCOS DE CUSTOMIZAÇÃO DE CONTROLO	BLOCOS DE CONFIGURAÇÃO CUSTOMIZÁVEL MAS DEFINIDA NO *SOFTWARE*

5.2.2 - Referências do ficheiro NCI

O ficheiro *.NCI – *Numerical Control Intermediate* – tal como já foi referido na descrição da estrutura do Mastercam X5, é gerado à semelhança das normas ISO 3592, ISO 4343 e ISO 6983-1/2/3 : Processamento em APT e geração de CLDATA. Como tal, a informação nele presente pode ser interpretada como CLDATA, ou, em inglês – *Cutter Location DATA*, também designada como CLFILE [1].

O CLFILE agrega toda a informação proveniente dos percursos calculados pelo *software* – juntamente com informações adicionais da preferência do utilizador – e dispõe-nas ao longo das suas linhas.

A cada linha está associada uma referência, descrita por um código numérico. Este código divide-se em grupos de tipo de informação, e a cada um está associada uma classe[47]. Logo, no CLFILE, a cada linha está associado um código que contém os seus parâmetros:

Tomando como exemplo um excerto de um ficheiro NCI:

```
( . . . )
20001
   25. FLAT ENDMILL
20002

20003

20004
234 10 1 0 25. 0. 0. 180. 234 234 610.8 305.4 305.4 1527 8 4
( . . . )
0
0 48.00563604 5.76380936 40. -2. 0
0
0 48.00563604 5.76380936 35. -2. 0
1
0 48.00563604 5.76380936 26.68713104 305.4 0
1
0 47.95767541 5.81919292 26.38196599 -1. 0
( . . . )
```

Nestes excertos de um CLFILE gerado pelo Mastercam, observam-se oito linhas de CLDATA. As quatro primeiras dizem respeito ao início do ficheiro. Os códigos 20001-4 significam, respetivamente, a designação dada à ferramenta, a designação/código do fabricante, a designação dada ao elemento de fixação, e os parâmetros da ferramenta (número da ferramenta, tipo de ferramenta, diâmetro da ferramenta, comprimento, etc…). As quatro últimas linhas são relativas a dois movimentos rápidos (código 0 na referência), e duas interpolações lineares (código 1). Os seus parâmetros dizem respeito a posicionamento, compensação de diâmetro, e velocidades de avanço.

[47] Resumidamente, em programação estruturada por objectos, o termo classe é usado para designar uma construção que, neste contexto, contém dados sobre si mesma.

5.2.3 – Requisitos operacionais da implementação de um pós-processador

Como se viu nas subsecções anteriores, existem alguns requisitos associados à configuração da máquina virtual. Este subsecção aborda, em particular, as exigências da configuração de um pós-processador.

A construção ou edição de um pós-processador requer amplo conhecimento do funcionamento do código de comando numérico, assim como informação completa do controlador da máquina, das suas capacidades e características. Tipicamente, essas informações encontram-se referenciadas nos manuais das máquinas e seus controladores (como será possível observar nas subsecções seguintes).

Deve existir também, por parte do utilizador, a capacidade de responder a diversas questões que se levantam, respeitantes à utilização do sistema CAM em causa.

No Mastercam X5, todo o trabalho com pós-processadores – à exceção das configurações diretamente acessíveis através dos menus do ficheiro de controlo – é insuficientemente suportado, tanto pela falta de documentação adequada, como pela falta de editores dedicados[48].

Como se apresenta na Figura 80, é necessário o conhecimento do modo de funcionamento do ficheiro CLFILE, para que o seu conteúdo possa ser lido adequadamente, e não dar azo a erros na geração do código de comando numérico.

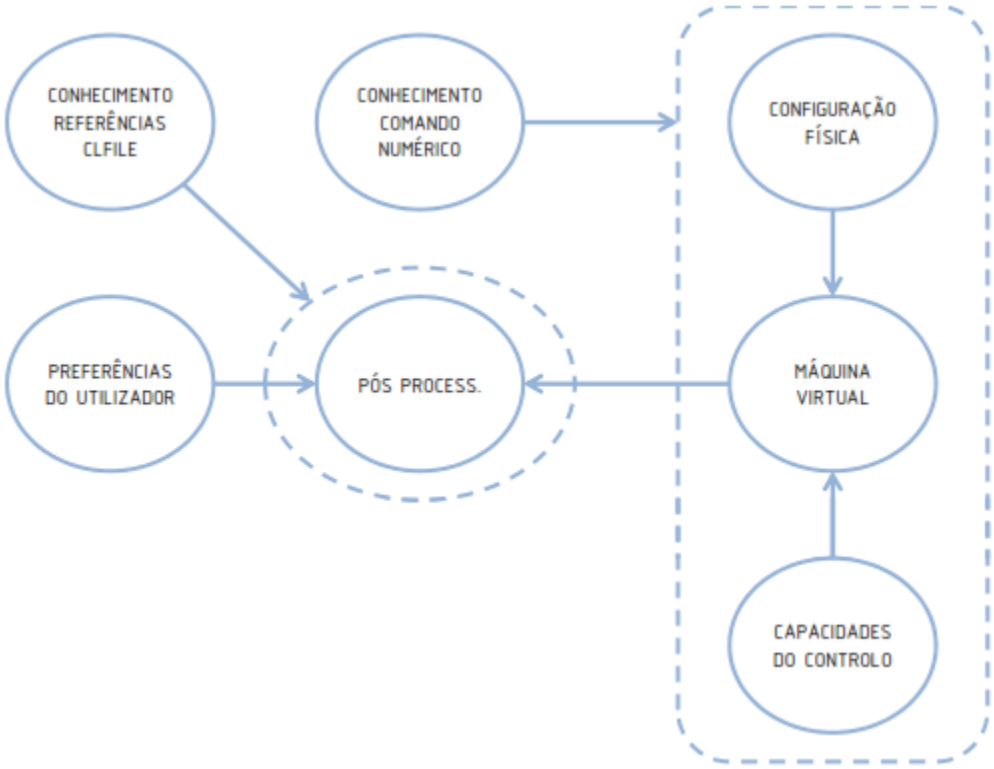
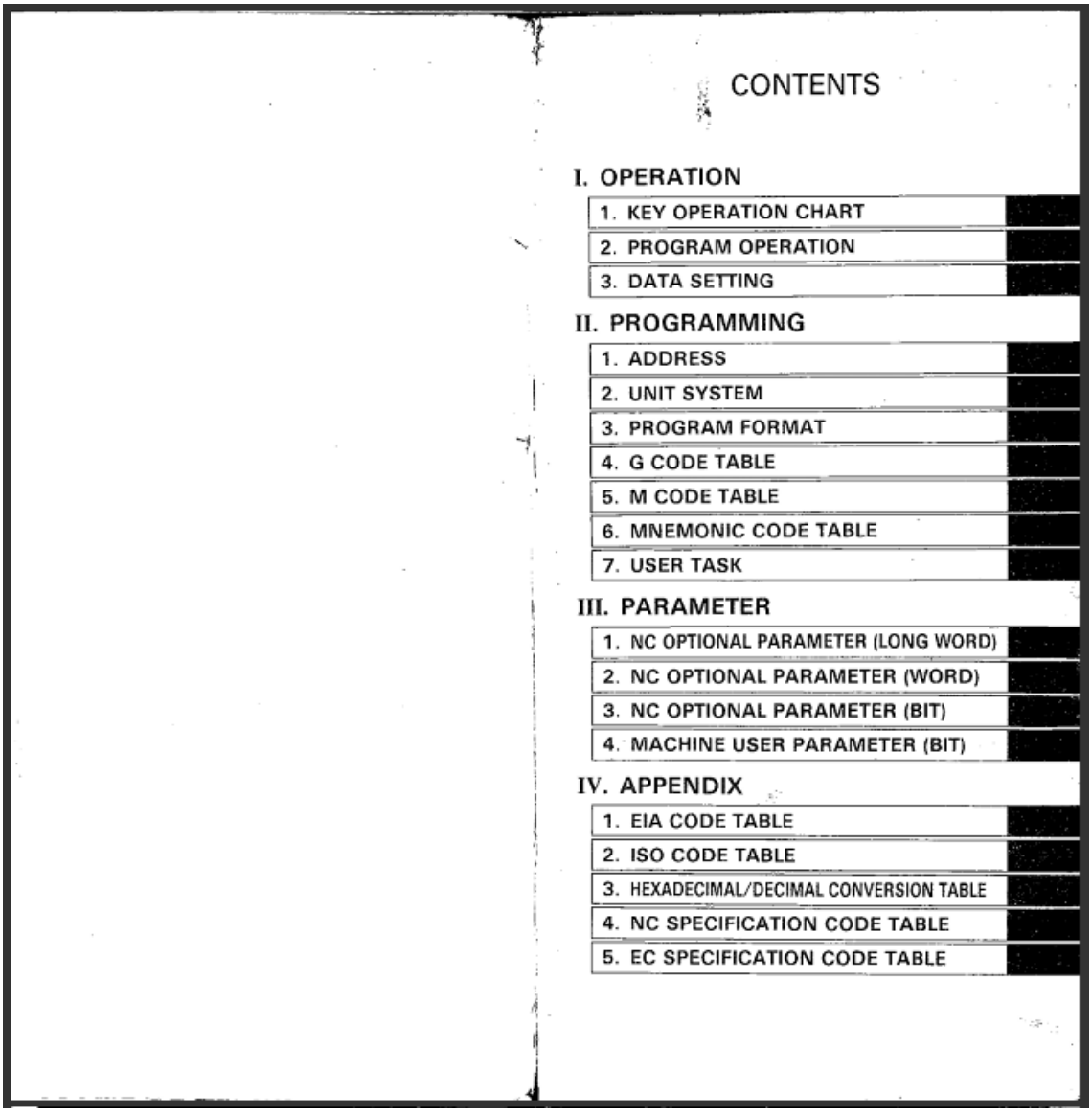


Figura 80 - Requisitos da programação de um pós-processador.

[48] Como um dos resultados e conclusões centrais deste trabalho, ir-se-á discutir com mais pormenor esta questão, inclusive apresentando soluções e estratégias para minorar as suas consequências.

A consulta de documentação fornecida pelos fabricantes da máquina-ferramenta e seu controlador permitem obter uma boa base das tolerâncias dimensionais do mesmo, assim como os códigos modais e auxiliares usados, *bytes* configuráveis, outros parâmetros opcionais, e operações embutidas (*canned cycles*[49]). O agrupamento dessas especificidades de utilização de controlo, pode, no caso do controlador OSP5020M (integrado no centro de maquinagem Okuma MC-40VA), ser encontrado no manual de operação da mesma, como se ilustra na Figura 81.

No próximo subsecção ir-se-ão reunir as informações pertinentes e aplicar este método ao caso específico desta dissertação (Okuma MC-40VA e controlador OSP5020M), configurando, para os seus efeitos, um pós-processador.


CONTENTS

I. OPERATION
1. KEY OPERATION CHART
2. PROGRAM OPERATION
3. DATA SETTING
II. PROGRAMMING
1. ADDRESS
2. UNIT SYSTEM
3. PROGRAM FORMAT
4. G CODE TABLE
5. M CODE TABLE
6. MNEMONIC CODE TABLE
7. USER TASK
III. PARAMETER
1. NC OPTIONAL PARAMETER (LONG WORD)
2. NC OPTIONAL PARAMETER (WORD)
3. NC OPTIONAL PARAMETER (BIT)
4. MACHINE USER PARAMETER (BIT)
IV. APPENDIX
1. EIA CODE TABLE
2. ISO CODE TABLE
3. HEXADECIMAL/DECIMAL CONVERSION TABLE
4. NC SPECIFICATION CODE TABLE
5. EC SPECIFICATION CODE TABLE

Figura 81 - Imagem retirada de um manual de utilização do controlador OSP5020M.

[49] *Canned cycles* é um termo inglês usado para designar operações integradas, que reúnem outras menores e mais elementares, mas que, através de uma definição de parâmetros específica, é mais simplesmente aplicada. A título de exemplo: ciclos de furação, ou de abertura de rosca.

5.3 – Características da máquina Okuma MC-40VA

Na subsecção anterior explicou-se a função do pós-processamento, a estrutura geral de um pós-processador, e os requisitos da sua implementação ou edição.

Neste ponto, de acordo com a sequência sugerida na subsecção 5.1, proceder-se-á à recolha de informações relativas à fresadora Okuma MC-40VA e ao controlador OSP5020M. Esta recolha estará então dividida em duas partes: uma relativa à Okuma, e outra ao controlador.

Na subsecção seguinte far-se-á uma proposta de configuração, utilizando as informações recolhidas e o conhecimento exposto na subsecção 5.2.

Capacidades de trabalho da Okuma MC-40VA

A Okuma MC-VA40 é um centro de maquinagem do tipo vertical de 3 eixos, que possui um armazém de ferramentas automático com 20 posições. As suas capacidades de trabalho encontram-se resumidas na Tabela 2.

Tabela 2 - Características físicas gerais do centro de maquinagem Okuma MC-VA40.

Dimensões da mesa	1000(X) x 410(Y)	mm
Volume de trabalho	762(X) x 410(Y) x 450(Z)	
Zero máquina (rel. mesa)	(0 , 0 , 200)	
Posição troca ferramentas	(381 , 205 , 650)	
Velocidade do *spindle*	10 - 6000	RPM
Velocidade de avanço máxima	20000 (X e Y)	mm/min
	15000 (Z)	
Velocidade de avanço em corte	1 - 4000 (X Y e Z)	
Capacidade do armazém	20 ferramentas	
Ø máx ferramenta (intercalada)	152	mm
Ø máx ferramenta	95	
Comprimento máx ferramenta	300	

As características apresentadas podem ser revistas, nos manuais de operação e/ou manutenção da máquina [24].

Relativamente à representação em 3D da máquina, para efeitos de deteção de colisões, retiraram-se informações básicas relativas ao atravancamento da mesma. Figura 82 [24].

Outras imagens presentes no manual foram utilizadas, de modo a representar componentes como a mesa, ou o *spindle.*

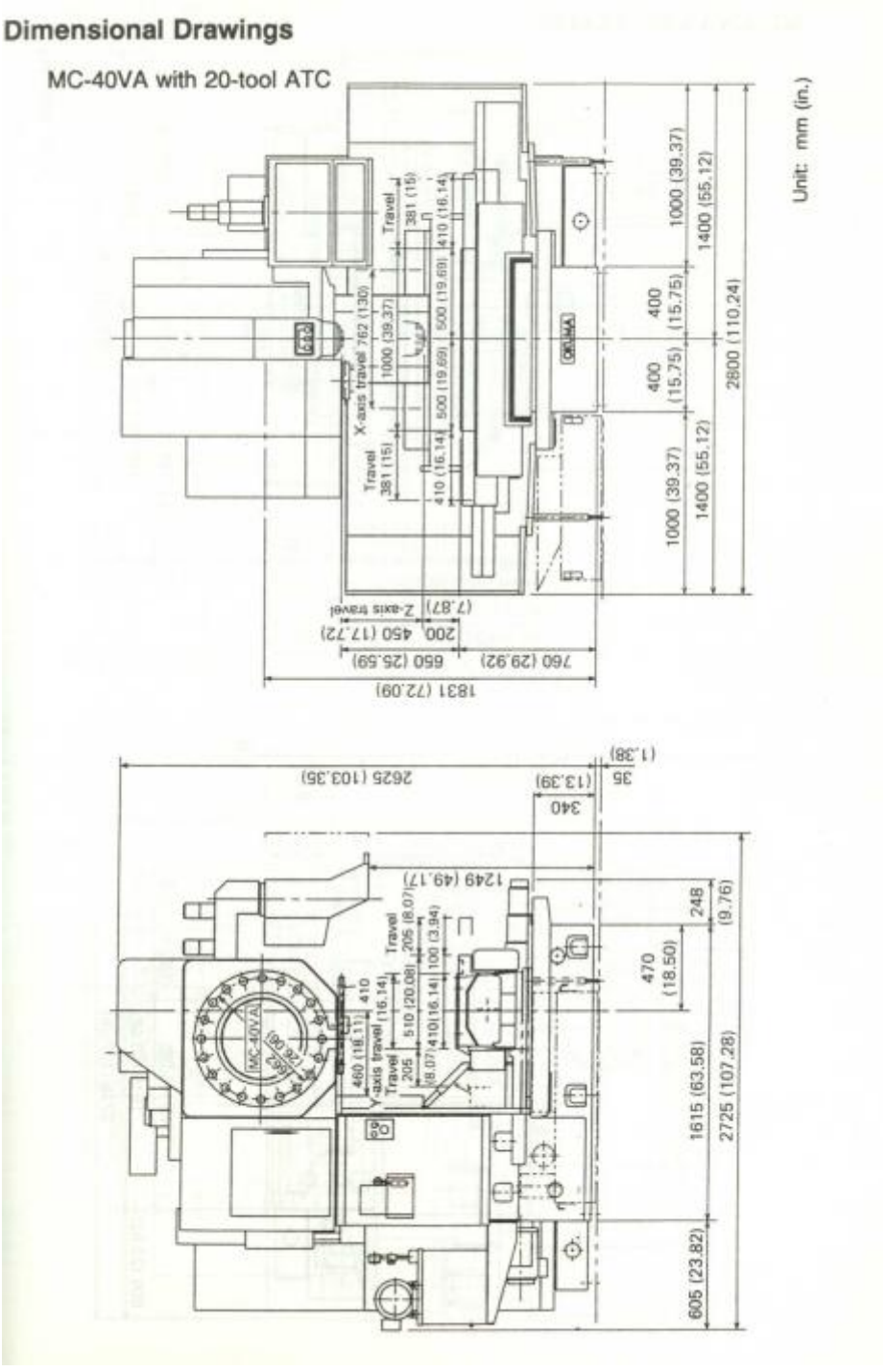


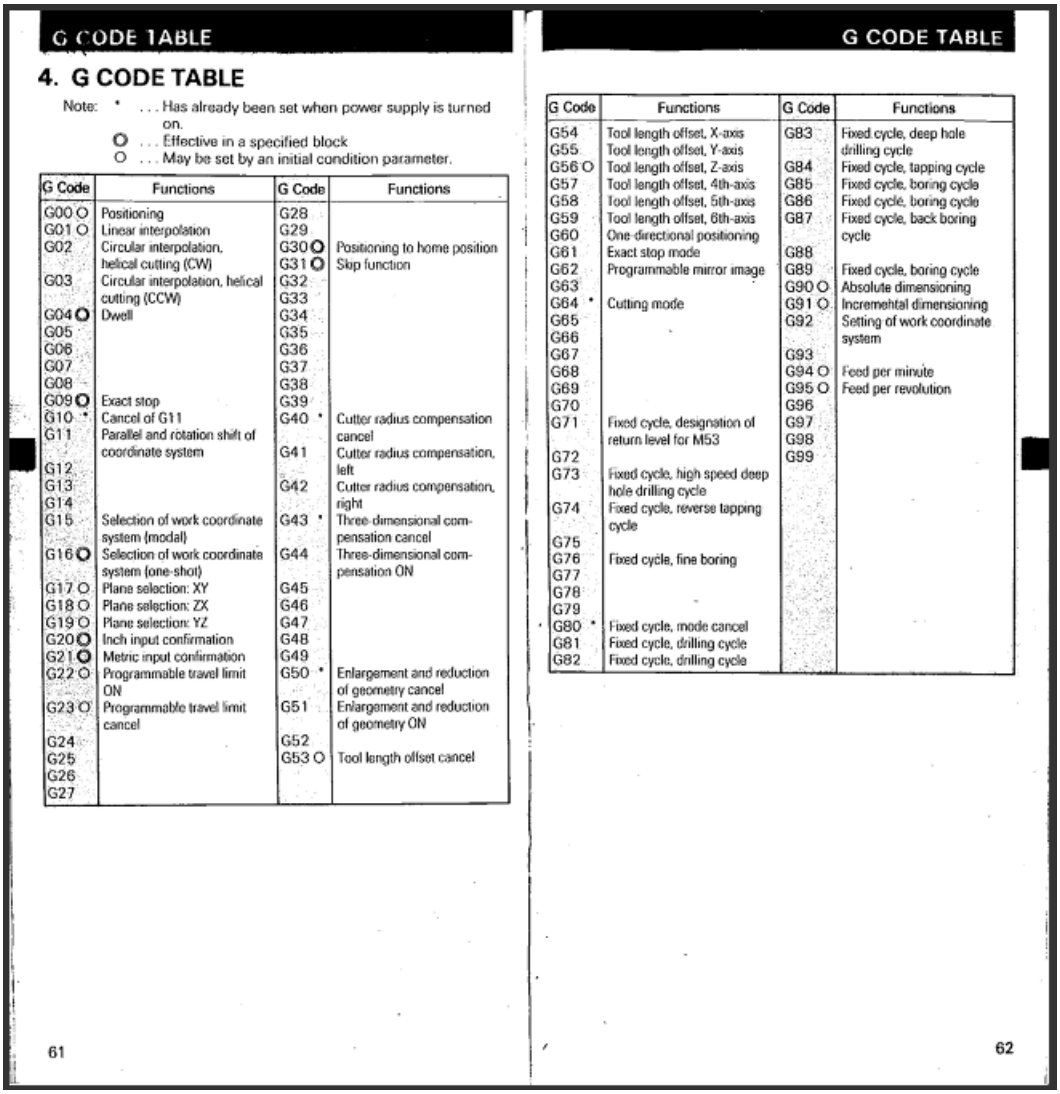
Figura 82 - Desenho esquemático das diversas dimensões de atravancamento, entre outras.

Características do controlador OSP5020M

O controlador OSP5020M é um controlador CNC, que, no contexto da interação com pós-processadores[50]:

- Traduz o código de comando numérico nas posições dos eixos;
- Regista a chamada de funções auxiliares e aciona os sistemas inerentes;
- Aplica tolerâncias na conversão de arcos e curvas.

Na Figura 83 mostra-se um exemplo da recolha de informações do controlador.



G CODE TABLE

4. G CODE TABLE

Note: * ... Has already been set when power supply is turned on.
O ... Effective in a specified block
O ... May be set by an initial condition parameter.

G Code	Functions	G Code	Functions
G00 O	Positioning	G28	
G01 O	Linear interpolation	G29	
G02	Circular interpolation, helical cutting (CW)	G30 O	Positioning to home position
		G31 O	Skip function
G03	Circular interpolation, helical cutting (CCW)	G32	
		G33	
G04 O	Dwell	G34	
G05		G35	
G06		G36	
G07		G37	
G08		G38	
G09 O	Exact stop	G39	
G10 *	Cancel of G11	G40 *	Cutter radius compensation cancel
G11	Parallel and rotation shift of coordinate system	G41	Cutter radius compensation, left
G12			
G13		G42	Cutter radius compensation, right
G14			
G15	Selection of work coordinate system (modal)	G43 *	Three-dimensional compensation cancel
G16 O	Selection of work coordinate system (one-shot)	G44	Three-dimensional compensation ON
G17 O	Plane selection: XY	G45	
G18 O	Plane selection: ZX	G46	
G19 O	Plane selection: YZ	G47	
G20 O	Inch input confirmation	G48	
G21 O	Metric input confirmation	G49	
G22 O	Programmable travel limit ON	G50 *	Enlargement and reduction of geometry cancel
G23 O	Programmable travel limit cancel	G51	Enlargement and reduction of geometry ON
G24		G52	
G25		G53 O	Tool length offset cancel
G26			
G27			

61

G CODE TABLE

G Code	Functions	G Code	Functions
G54	Tool length offset, X-axis	G83	Fixed cycle, deep hole drilling cycle
G55	Tool length offset, Y-axis		
G56 O	Tool length offset, Z-axis	G84	Fixed cycle, tapping cycle
G57	Tool length offset, 4th-axis	G85	Fixed cycle, boring cycle
G58	Tool length offset, 5th-axis	G86	Fixed cycle, boring cycle
G59	Tool length offset, 6th-axis	G87	Fixed cycle, back boring cycle
G60	One-directional positioning		
G61	Exact stop mode	G88	
G62	Programmable mirror image	G89	Fixed cycle, boring cycle
G63		G90 O	Absolute dimensioning
G64 *	Cutting mode	G91 O	Incremehtal dimensioning
G65		G92	Setting of work coordinate system
G66			
G67		G93	
G68		G94 O	Feed per minute
G69		G95 O	Feed per revolution
G70		G96	
G71	Fixed cycle, designation of return level for M53	G97	
		G98	
G72		G99	
G73	Fixed cycle, high speed deep hole drilling cycle		
G74	Fixed cycle, reverse tapping cycle		
G75			
G76	Fixed cycle, fine boring		
G77			
G78			
G79			
G80 *	Fixed cycle, mode cancel		
G81	Fixed cycle, drilling cycle		
G82	Fixed cycle, drilling cycle		

62

Figura 83 - Imagem retirada de um manual de utilização do controlador OSP5020M – códigos modais.

Da Tabela 3 à Tabela 10 resumem-se os códigos e funções mais relevantes do controlador, referindo o estado da sua implementação, em contraste com o pós-processador original.

[50] Refira-se que toda a estrutura, funcionamento, e especifidades do controlador constituem por si só um estudo que vai para além do âmbito desta dissertação. Por esse motivo enquadram-se apenas aspetos relacionadas com a mesma.

Tabela 3 - Códigos modais e funções auxiliares usados no controlador OSP5020M. | parte 1 |

Código modal OSP5020M	Significado (no OSP5020M)	Pós-processador original (FANUC)	Anotações
G00	Movimento rápido	G00	Idêntico - não requer alterações.
G01	Interpolação linear (corte)	G01	
G02	Interpolação circular CW	G02	
G03	Interpolação circular CCW	G03	
G04	Pausa/permanência	G04	
G09	Paragem exata pontual	G09	
G10	Cancelar G11		No controlador FANUC estes códigos são usados para permitir (G10) ou cancelar (G11) a gravação de dados; Implementação opcional.
G11	Transformação de sistemas de coordenadas		
G15	Seleção de sistema de coordenadas modal	G53	O G53 é não modal, em coordenadas absolutas, e com uma posição (semelhante ao G92); requer implementação
G16	Seleção de sistema de coordenadas pontual		Não tem utilidade prática na programação em CN com a Okuma MC-40VA.

Tabela 4- Códigos modais e funções auxiliares usados no controlador OSP5020M. | parte 2 |

Código modal OSP5020M	Significado (no OSP5020M)	Pós-processador original (FANUC)	Anotações
G17	Plano XY	G17	Idêntico - não requer alterações.
G18	Plano YZ	G18	
G19	Plano XZ	G19	
G20	Unidades no sistema inglês	G20	
G21	Unidades no sistema métrico	G21	
G22	Limite de eixos ON		Não se encontra uma correspondência, embora se suponha que haja uma; requer implementação.
G23	Limite de eixos OFF		
G30	Posição CASA	G28/G30	Requer alteração ao código usado
G31	Ignorar a função	G31	Idêntico - não requer alterações.
G40	Cancelar compensação diâmetro	G40	

Tabela 5 - Códigos modais e funções auxiliares usados no controlador OSP5020M. | parte 3 |

Código modal OSP5020M	Significado (no OSP5020M)	Pós-processador original (FANUC)	Anotações
G41	Compensação diâmetro à esquerda	G41	Idêntico - não requer alterações.
G42	Compensação diâmetro à direita	G42	
G43	Cancelar compensação 3D		No controlador FANUC estes códigos são usados para adicionar/retirar *offset* de ferramenta; Implementação opcional.
G44	Compensação 3D		
G50	Cancelar G51	G50	Idêntico - não requer alterações.
G51	Escala da peça (aumento e redução)	G51	
G53	Cancelar offset ferramenta	G49	Requer alteração ao código usado
G54	Offset ferramenta direção X		No controlador FANUC estes códigos são usados para definir várias posições de sistemas de coordenadas de trabalho (G54-59); requer alteração da rotina no pós-processador e correção dos códigos.
G55	Offset ferramenta direção Y		
G56	Offset ferramenta direção Z		

Tabela 6 - Códigos modais e funções auxiliares usados no controlador OSP5020M. | parte 4 |

Código modal OSP5020M	Significado (no OSP5020M)	Pós-processador original (FANUC)	Anotações
G57	Offset ferramenta direção 4º eixo		No controlador FANUC estes códigos são usados para definir várias posições de sistemas de coordenadas de trabalho (G54-59); requer alteração da rotina no pós-processador e correção dos códigos.
G58	Offset ferramenta direção 5º eixo		
G59	Offset ferramenta direção 6º eixo		
G60	Posicionamento unidirecional		Sem correspondência; Implementação opcional.
G61	Paragem exata modal	G61	Idêntico - não requer alterações.
G62	Espelhamento ON/OFF		Na FANUC serve para controlar o *cornering*[51]; não requer alteração - operação definida pelo programa
G64	Cancelar paragem exata modal	G64	Idêntico - não requer alterações.
G71	Posição de retorno para ciclos de furação	G99	Requer alteração ao código usado
G73	Ciclo de furação a alta velocidade	G73	Idêntico - não requer alterações - com exceção da definição da posição de retorno (G99 para G73), feita externamente.
G74	Ciclo de roscagem inverso	G74	

[51] *Cornering*, do inglês “fazer cantos”, é usado para designar o ato de contornar um canto. Neste contexto diz respeito à forma como as ferramentas contornam os cantos.

Tabela 7 - Códigos modais e funções auxiliares usados no controlador OSP5020M. | parte 5 |

Código modal OSP5020M	Significado (no OSP5020M)	Pós-processador original (FANUC)	Anotações
G76	Ciclo de broqueamento fino	G76	Idêntico - não requer alterações - com exceção da definição da posição de retorno (G99 para G73), feita externamente.
G80	Cancelar ciclos embebidos	G80	
G81	Ciclo de furação	G81	
G82	Ciclo de furação com permanência[52]	G82	
G83	Ciclo de furação de alta profundidade	G83	
G84	Ciclo de roscagem	G84	
G85	Ciclo de broqueamento com retorno em corte	G85	
G86	Ciclo de broqueamento com retorno rápido	G86	
G87	Ciclo de broqueamento invertido	G87	
G89	Ciclo de broqueamento com permanência	G89	

[52] Permanência é um termo usado para designar um ponto da operação em que a ferramenta permanece, numa posição, durante um espaço de tempo definido.

Tabela 8 - Códigos modais e funções auxiliares usados no controlador OSP5020M. | parte 6 |

Código modal OSP5020M	Significado (no OSP5020M)	Pós-processador original (FANUC)	Anotações
G90	Programação modo absoluto	G90	Idêntico - não requer alterações.
G91	Programação modo incremental	G91	
G92	Utilização de coordenadas-peça	G92*/G52	
G94	Modo avanço por minuto	G94	
G95	Modo avanço por revolução	G95	
Função auxiliar OSP5020M	Significado (no OSP5020M)	Pós-processador original (FANUC)	Anotações
M00	Paragem de programa	M00	Idêntico - não requer alterações.
M01	Paragem opcional	M01	
M02	Fim de programa	M02	
M03	Rotação *spindle* CW	M03	

Tabela 9 - Códigos modais e funções auxiliares usados no controlador OSP5020M. | parte 7 |

Função auxiliar OSP5020M	Significado (no OSP5020M)	Pós-processador original (FANUC)	Anotações
M04	Rotação spindle CCW	M04	Idêntico - não requer alterações.
M05	Paragem do spindle	M05	
M06	Troca de ferramenta	M06	
M07	Refrigeração/lubrificação por neblina de óleo	M07	
M08	Refrigeração/lubrificação por inundação	M08	
M09	Desativação de M07, M08, M12, M50, M51	M09	
M12	Sopragem de ar para limpeza de rebarba		Pouco usado; Implementação opcional.
M30	Fim de fita perfurada	M30	Idêntico - não requer alterações.
M32	Abertura de porta de protecção		Pouco usado; Implementação opcional.
M33	Fecho de porta de protecção		

Tabela 10 - Códigos modais e funções auxiliares usados no controlador OSP5020M. | parte 8 |

Função auxiliar OSP5020M	Significado (no OSP5020M)	Pós-processador original (FANUC)	Anotações
M40	Selecção de gamas de potência/binário/velocidade do eixo	M40	Não será implementado.
M41	Selecção de gamas de potência/binário/velocidade do eixo	M41	Não será implementado.
M42		M42	
M43		M43	
M50	Refrigeração na ferramenta em baixa pressão	M08	Não existe sistema de refrigeração através da ferramenta; não será implementado.
M51	Refrigeração na ferramenta em alta pressão		
M70	Troca manual de ferramenta		Não será implementado..
M87	Ativação de neblina de óleo e sopragem		
CALL OSUB	Chamada de subrotina	M98	Requer alteração ao código usado.
RTS	Retorno a programa principal	M99	

O controlador OSP5020M possui ainda alguns aspetos relevantes para a configuração do Mastercam X5, no sistema métrico, que devem ser considerados:

- Resolução da posição linear: 0,001 mm (parametrizável);
- Resolução da posição angular: 0,001º (parametrizável);
- Resolução da definição de velocidade de avanço/tempo: 1 ou 0,1 mm/min (parametrizável) ;
- Resolução da velocidade de rotação do *spindle: 1 RPM;*
- Resolução da velocidade de avanço/rotação: 0,01 ou 0,001 mm/rot (parametrizável).

Existem outras funções do controlador OSP5020M, tais como a repetição em padrão de uma operação, a abertura de *pockets*, ou o facejamento pré-definido. No entanto, no âmbito do estudo da utilização de uma oferta de CAM comercial em particular, e porque o equivalente no *software* dessas funções é significativamente mais completo, escolheu-se deixá-las a cargo do mesmo.

Existem ainda outras variáveis e parâmetros acessíveis através da consola da Okuma MC-40VA, mas porque fazem parte do funcionamento da máquina *in situ*, não serão abordados neste trabalho.

5.4 – Implementação da máquina virtual

De acordo com as informações acerca das capacidades da máquina e controlador, e observando as necessidades de edição do pós-processador, estão reunidas as condições para elaborar um plano de configuração do programa para estes equipamentos:

1. A máquina virtual criada deve conter as informações das capacidades de trabalho (volume de trabalho), assim como as gamas das velocidades de avanço em corte, avanço rápido, e rotação do *spindle*. Deve também conter a posição de troca de ferramenta, a capacidade do armazém, e os tempos de troca de ferramenta;
2. O controlador associado à máquina virtual deve conter as configurações de resolução das posições lineares e angulares, assim como da utilização de alguns códigos modais e auxiliares. Neste caso aplicam-se também algumas opções relativas às capacidades tecnológicas do OSP5020M/Okuma MC-40VA:
    a. Maquinagem a alta velocidade – não é permitido;
    b. Interpolação em hélice – é permitido;
    c. Utilização de sub-rotinas – é permitido;
    d. Outras opções relativas à utilização de informação da máquina virtual;
    e. Parâmetros de comunicação;
    f. Outros.
3. Relativamente ao pós-processador, este conterá a alteração direta dos códigos modais, auxiliares, e funções especiais, que diferem do pós-processador pré-definido. Aqui serão implementadas outras funções para suportar os modos de funcionamento

diferentes dos códigos específicos do OSP5020M. Outras implementações introduzidas no pós-processador são:

a. Rotinas de verificação de programação fora dos limites do volume de trabalho da máquina;
b. Rotinas de extração de informação do comprimento da ferramenta;
c. Rotinas de extração de informação da máquina virtual para melhor utilização no pós-processador.

No próximo subsecção ilustrar-se-ão as alterações, assim como o código produzido com recurso ao novo pós-processador, usando para o efeito os mesmos modelos apresentados no capítulo anterior.

Em anexo encontra-se todo o código do pós-processador.

Comece-se pelas definições gerais da máquina virtual. Na Figura 84 mostra-se a sua seleção, no gestor de máquinas virtuais. A máquina deve ser colocada numa pasta pré-designada para o efeito, ou noutra à escolha do utilizador – desde que seja definida essa pasta na configuração da interface do Mastercam. Aqui a máquina é inserida na barra de comandos do programa, passando a fazer parte do próprio menu.

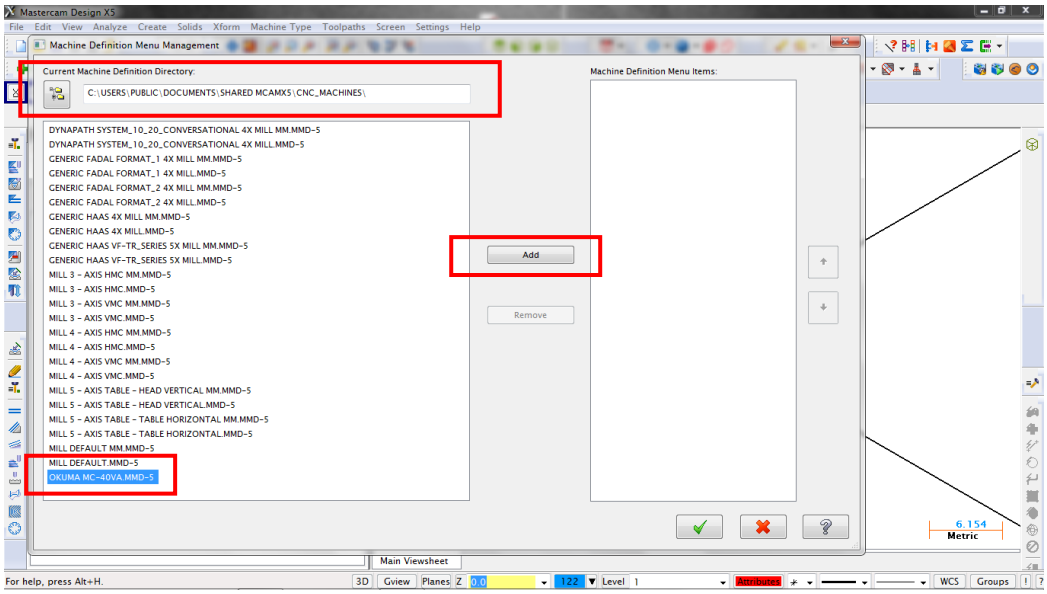


Figura 84 - Escolha da máquina virtual para posterior configuração.

Na Figura 85, é possível ver essa inclusão.

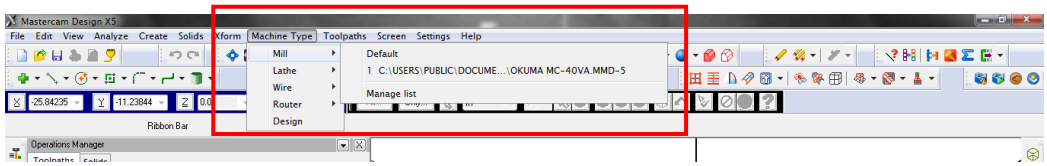


Figura 85 - Visualização da máquina na lista de escolhas possíveis.

Tendo já sido inserida a máquina no modelo (*mcx-5), pode-se desenvolver os passos seguintes:

1. Configurar a árvore de componentes da máquina virtual;
2. Definir os seus parâmetros gerais;
3. Definir um ficheiro de controlo,
4. Configurar o ficheiro de controlo para poder usar as edições do pós-processador, escolhido também neste ponto (a apresentação do pós-processador será feita na subsecção seguinte. A definição do pós-processador é feita aqui por razões de consistência do processo);
5. Mostra-se a máquina virtual já modelada em 3D;
6. Por fim será apresentado o programa de simulação da máquina virtual.

Na Figura 86 mostra-se como se acede ao menu de configuração da máquina virtual. Apesar de ser possível aceder a esta definição através dos parâmetros da máquina já inserida na árvore das operações (à esquerda), essa seleção diria respeito à instância da máquina virtual que foi implementada no modelo (como se explicou no capítulo 3). Acedendo à configuração da máquina virtual através do menu "*Machine Definition Manager*", é possível alterar a versão global desta máquina virtual (a OKUMA MC-40VA.mmd-5). Depois será possível substituir automaticamente a versão que se encontra no modelo pela versão mais recente.

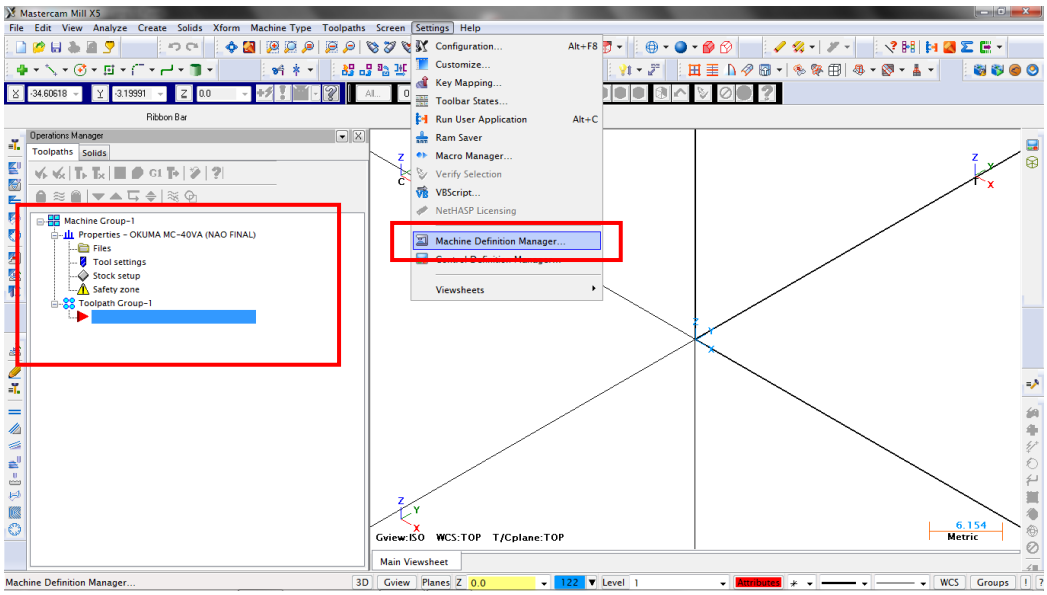


Figura 86 - Inicialização do menu de definições da máquina virtual.

Na Figura 87 mostra-se o menu inicial da máquina virtual, onde ainda não está inserida a informação relativa ao ficheiro de controlo a utilizar. Consequentemente, a máquina não dispõe ainda de pós-processador. Para este exemplo, a localização das base de dados de ferramentas, operações, e parâmetros pré-definidos das mesmas, ficarão por definir. Embora faça parte do menu da máquina virtual, a sua definição não é crucial para o seu bom funcionamento. No passo seguinte deve-se entrar no menu de configurações gerais.

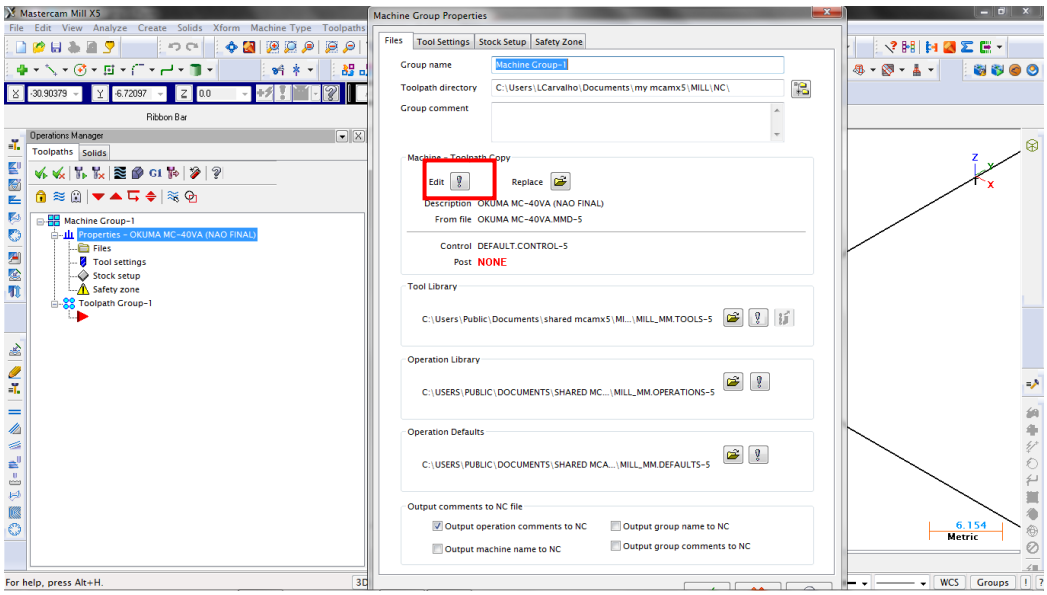


Figura 87 - Menu inicial de configuração da máquina virtual.

Entrando nesse menu, como se pode ver na Figura 88, dispõe-se vários elementos. À direita está representada a árvore de componentes (já de acordo com a Okuma MC-40VA) e à esquerda encontra-se a lista de componentes (caso se queira adicionar algum). Em cima acede-se a outros menus, nomeadamente de outras opções gerais (não relacionadas com componentes) e definições do ficheiro de controlo, entre outras.

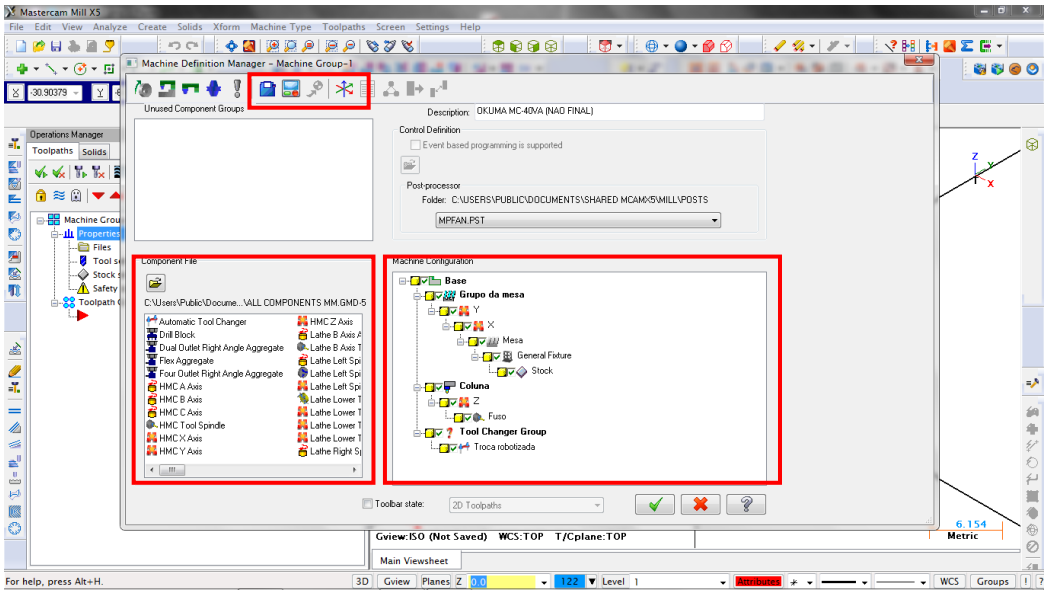


Figura 88 - Menu de definições gerais da máquina virtual - lista de componentes e árvore de componentes.

Exemplificando a definição de um componente, mostra-se na Figura 89, a inserção de dados relativamente aos limites (Y∈[-205, 205]), posição neutra (Y=0), e velocidades máximas de avanço rápido do eixo Y (VMARy=20000 mm/min). Este menu inclui outras opções, mas que não serão descritas.

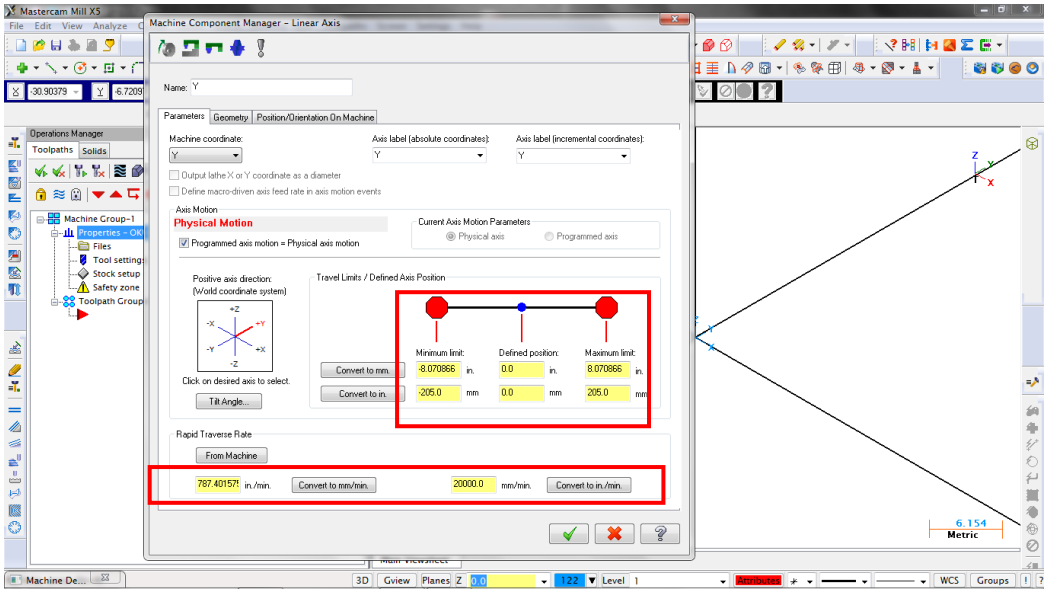


Figura 89 - Parametrização de um componente em particular - eixo Y.

Definidos os eixos, pode-se passar à parametrização do *spindle*. Na Figura 90 observa-se este processo. Aqui definem as velocidades mínimas e máximas, assim como a orientação da ferramenta, e o seu ponto de fixação[53].

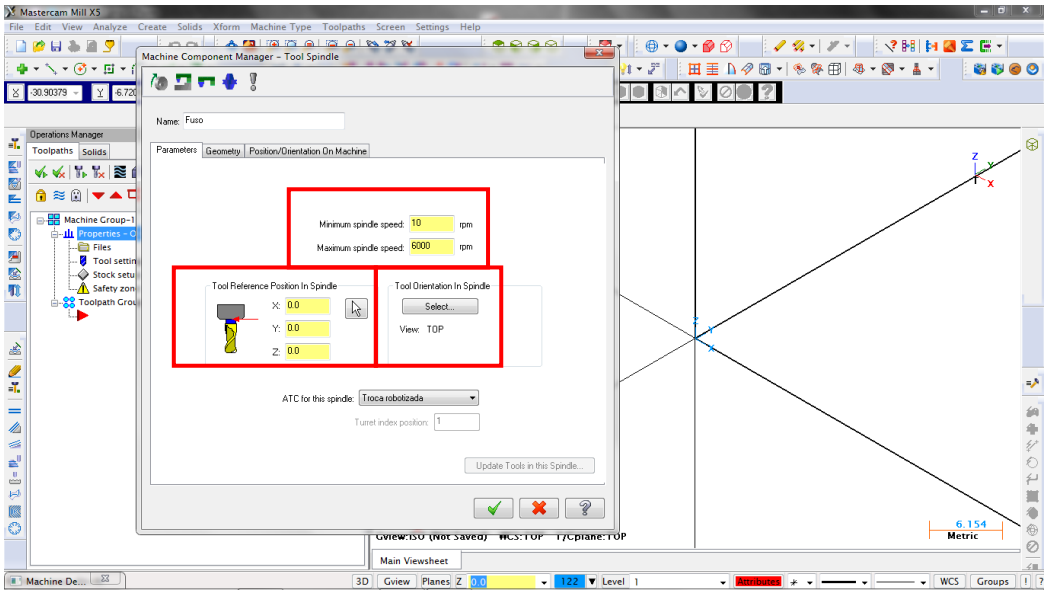


Figura 90 - Parametrização de um componente em particular - *spindle*.

[53] De notar que, para efeitos de modelação 3D, o *spindle* deve ser modelado de tal forma que este ponto de fixação coincida com o zero de todos os eixos (i.e. (x,y,z)=(0,0,0)), contactando com a mesa, algo que na realidade não acontece. Isto significa que o modelo 3D da máquina deve estar distorcido até ser inserido no Mastercam e, devido aos parâmetros dos eixos, ser aí corrigido. Esta configuração será possível de observar nas próximas páginas.

Na Figura 91 mostra-se outro exemplo de configuração, definindo o número de posições no armazém de ferramentas, e o tempo despendido com o seu funcionamento. Esta informação será usada posteriormente em simulações, para efeitos de estimativa de tempos de ciclo.

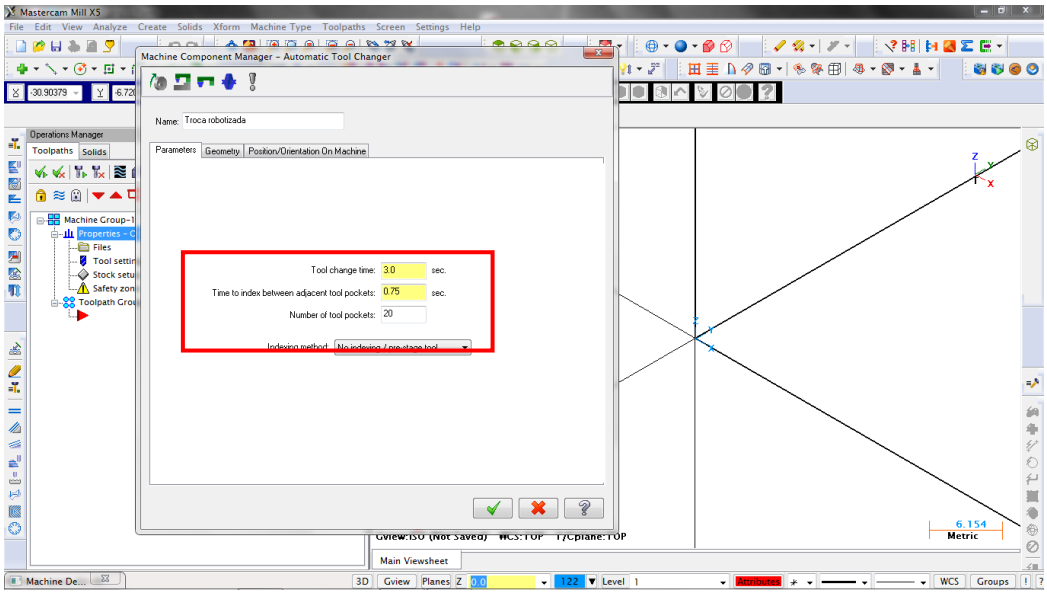

Machine Component Manager - Automatic Tool Changer
Name: Troca robotizada
Parameters
Geometry
Position/Orientation On Machine
Tool change time: 3.0 sec.
Time to index between adjacent tool pockets: 0.75 sec.
Number of tool pockets: 20
Main Viewsheet

Figura 91 - Parametrização de um componente em particular - armazém de ferramentas robotizado.

De seguida é necessário configurar o tipo de combinação de eixos, algo que é crucial no bom funcionamento do programa. Este ponto, como se pode observar na Figura 92, é feito definindo uma combinação composta pelos 3 eixos: X, Y e Z. Isto habilita a máquina virtual a usar operações 3D.

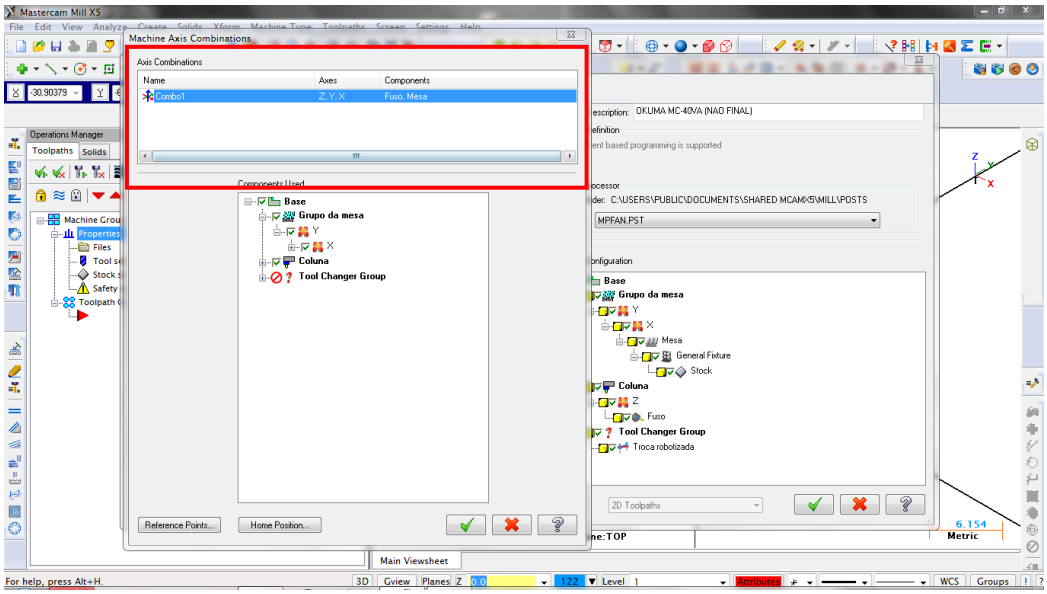


Figura 92 - Configuração da combinação de eixos.

Ainda neste menu define-se a posição casa (ou, do inglês, *home*). De acordo com o manual da Okuma MC-40VA, a posição casa, nesta situação, pode ser interpretada como a posição de troca de ferramenta. Como a Okuma é uma máquina em que a ferramenta apenas se desloca ao longo do eixo Z – e mais uma vez de acordo com o manual – essa posição de troca de ferramenta corresponderá ao afastamento máximo da mesa de trabalho, ou seja, às coordenadas (-381,205,650). Na Figura 93 mostra-se essa configuração.

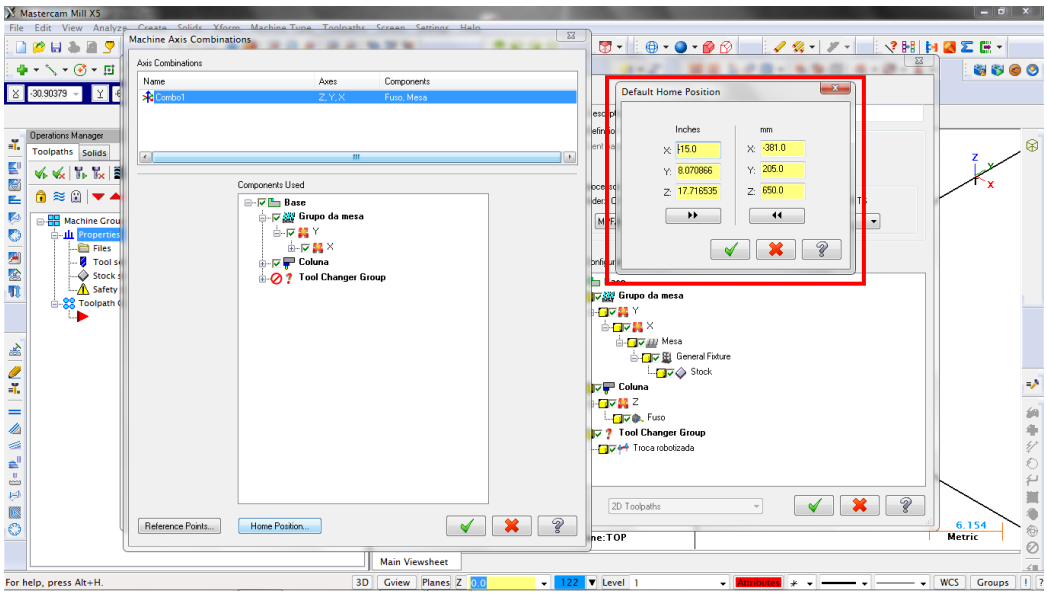


Figura 93 - Configuração da posição de troca de ferramenta.

O último passo das configurações gerais da máquina virtual, é através do menu "*General Machine Parameters*". Aqui definem-se alguns parâmetros gerais adicionais. Dentro desses, têm-se por exemplo como se mostra na Figura 94, as velocidades mínima e máxima de avanço em corte (1 mm/min e 4000 mm/min, respetivamente).

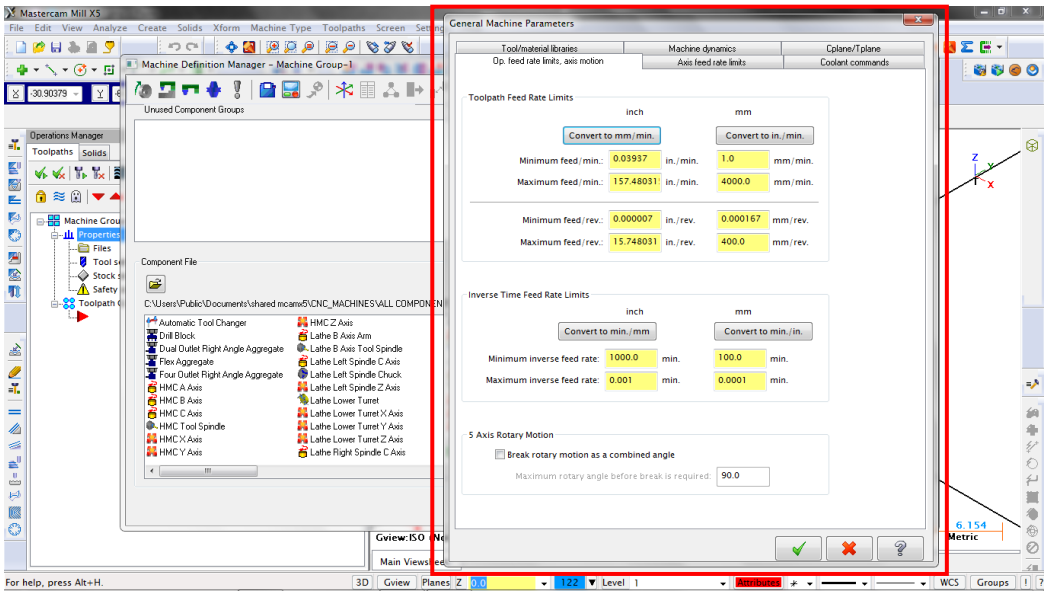

Mastercam Mill X5
General Machine Parameters
Machine Definition Manager - Machine Group-1
Toolpath Feed Rate Limits
Minimum feed/min.. 0.03937 in./min. 1.0 mm/min.
Maximum feed/min.. 157.48031 in./min. 4000.0 mm/min.
Minimum feed/rev.. 0.000007 in./rev. 0.000167 mm/rev.
Maximum feed/rev.. 15.748031 in./rev. 400.0 mm/rev.
Inverse Time Feed Rate Limits
Minimum inverse feed rate: 1000.0 min. 100.0 min.
Maximum inverse feed rate: 0.001 min. 0.0001 min.
5 Axis Rotary Motion
Break rotary motion as a combined angle
Maximum rotary angle before break is required: 90.0

Figura 94 - Parâmetros gerais comuns.

Apesar de ser possível aceder ao ficheiro controlo através das definições da máquina virtual associadas ao modelo, para configurar o controlador deve-se ir antes do menu designado "*Control Definition Manager*" (Figura 95).

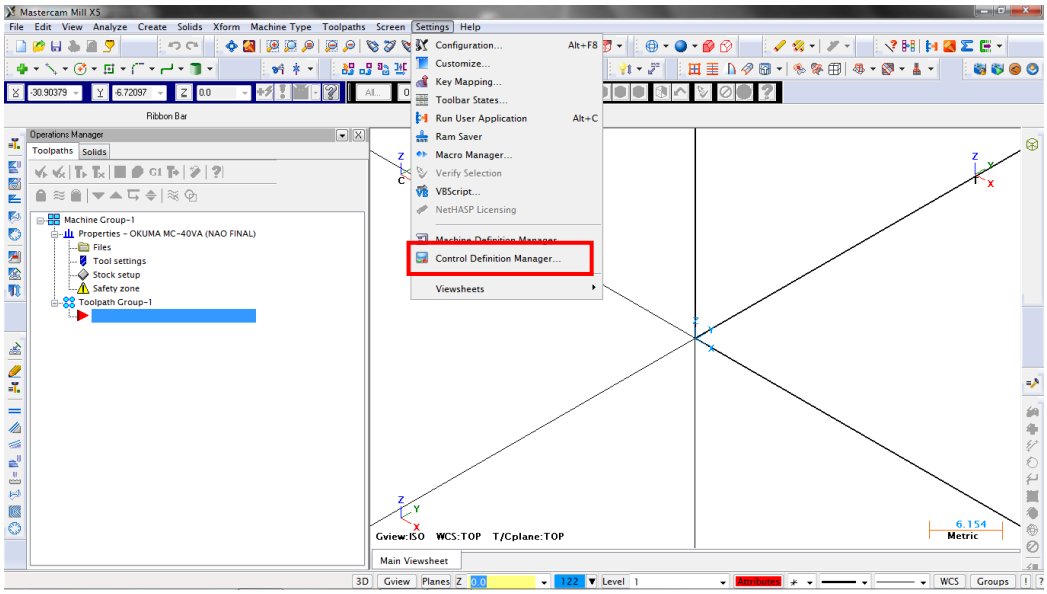


Figura 95 - Início da configuração das definições do ficheiro controlo.

O primeiro passo na definição do ficheiro de controlo é a sua escolha. Na Figura 96 mostra-se o carregamento de um ficheiro de controlo, e a lista de opções que o mesmo fornece. De seguida ter-se-á de definir algumas opções que permitam a utilização do pós-processador editado para esta máquina virtual.

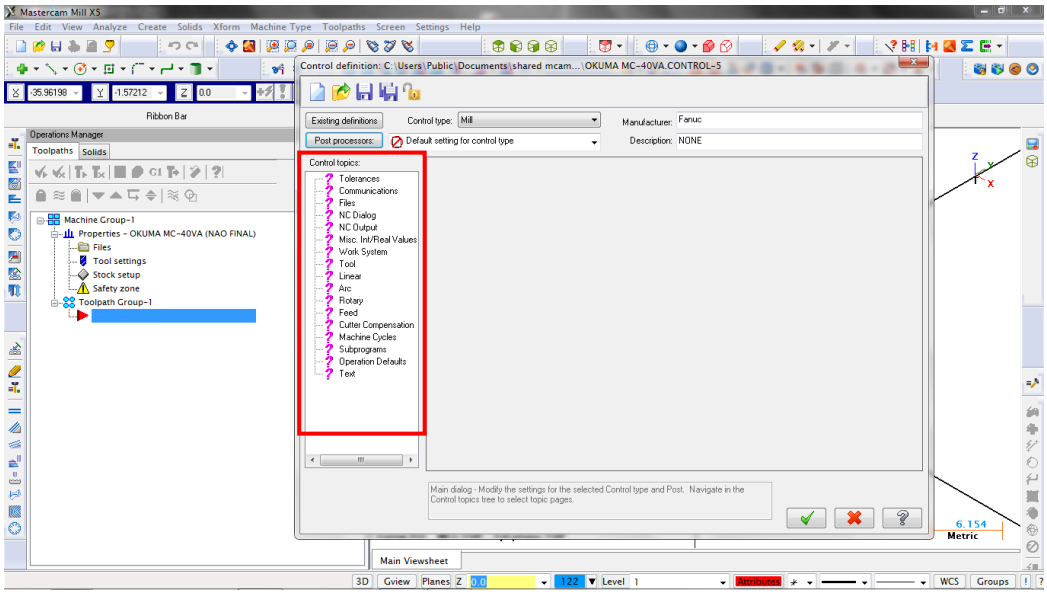


Figura 96 - Lista de tipos de opções associadas ao ficheiros de controlo.

É possível observar, na Figura 97, algumas opções relacionadas com a resolução do controlador, tolerâncias e desvios. O manual de utilização do Mastercam X5 aconselha a não edição destes parâmetros alertando para o facto que, estas tolerâncias estarão já definidas no próprio controlador de CN, aquando da leitura do código de CN produzido.

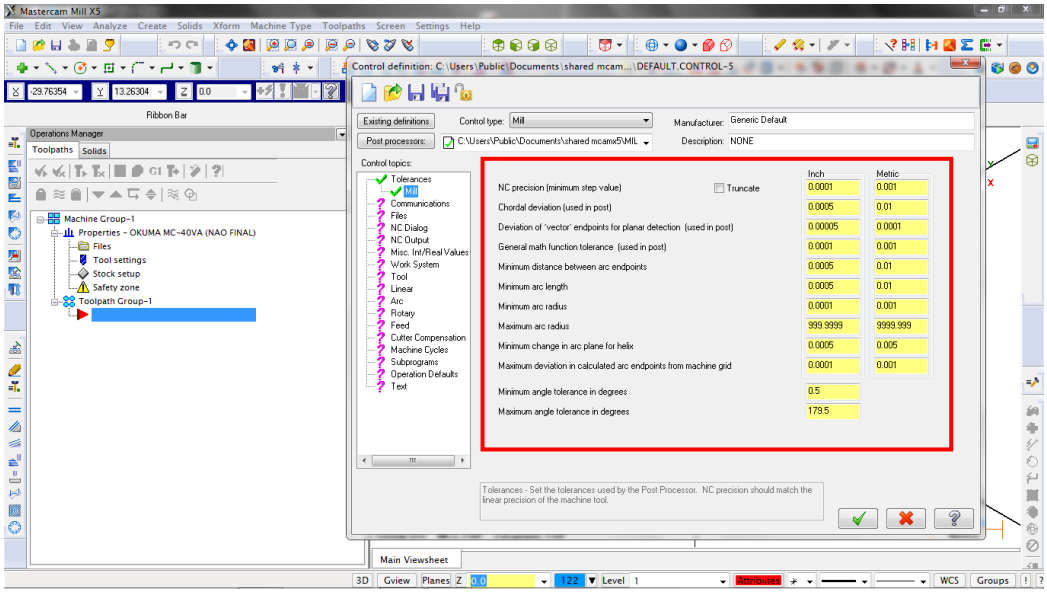


Figura 97 - Opções relacionadas com tolerâncias e resoluções do controlador OSP5020M.

Omitindo alguns passos intermédios, que não requerem especial atenção, atinge-se um que deve ser exemplificado. Na Figura 98 mostra-se o meu de valores customizáveis. Os valores customizáveis são, como o nome indica, variáveis que são definidas no menu do ficheiro de controlo, e têm influência direta no pós-processador. São portanto muito úteis, na medida em que permitem controlar algumas funções especiais criadas pelo utilizador e implementadas no pós-processador. Neste caso mostra-se alguns valores já criados.

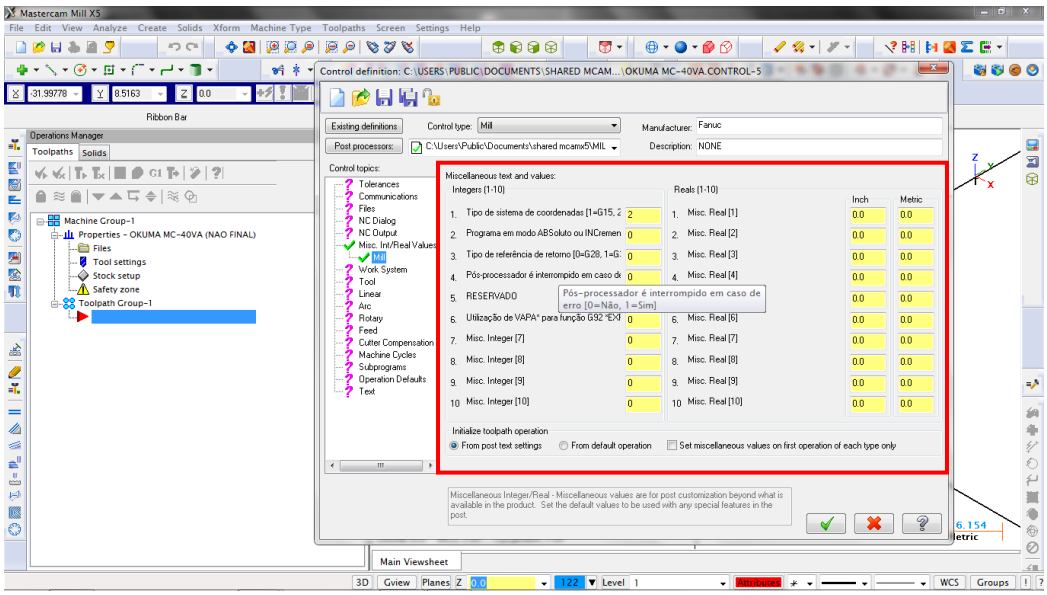

Mastercam Mill X5
Existing definitions
Control type: Mill
Manufacturer: Fanuc
Post processors:
Description: NONE
Control topics:
Miscellaneous text and values:
Integers (1-10)
Reals (1-10)
Tipo de sistema de coordenadas [1=G15, 2
Programa em modo ABSoluto ou INCremen
Tipo de referência de retorno [0=G28, 1=G
Pós-processador é interrompido em caso de
RESERVADO
Utilização de VAPA* para função G92 *EX
Misc. Integer [7]
Misc. Integer [8]
Misc. Integer [9]
Misc. Integer [10]
Misc. Real [1]
Misc. Real [2]
Misc. Real [3]
Misc. Real [4]
Misc. Real [6]
Misc. Real [7]
Misc. Real [8]
Misc. Real [9]
Misc. Real [10]
Pós-processador é interrompido em caso de erro [0=Não, 1=Sim]
Initialize toolpath operation
From post text settings
From default operation
Set miscellaneous values on first operation of each type only
Miscellaneous Integer/Real - Miscellaneous values are for post customization beyond what is available in the product. Set the default values to be used with any special features in the post.
Main Viewsheet

Figura 98 - Lista de valores customizáveis.

Para terminar a fase de configuração do ficheiro de controlo, deve-se por fim selecionar o pós-processador que, através deste mesmo ficheiro de controlo, será utilizado pela máquina virtual na tradução dos percursos gerados. Essa escolha evidencia-se na Figura 99.

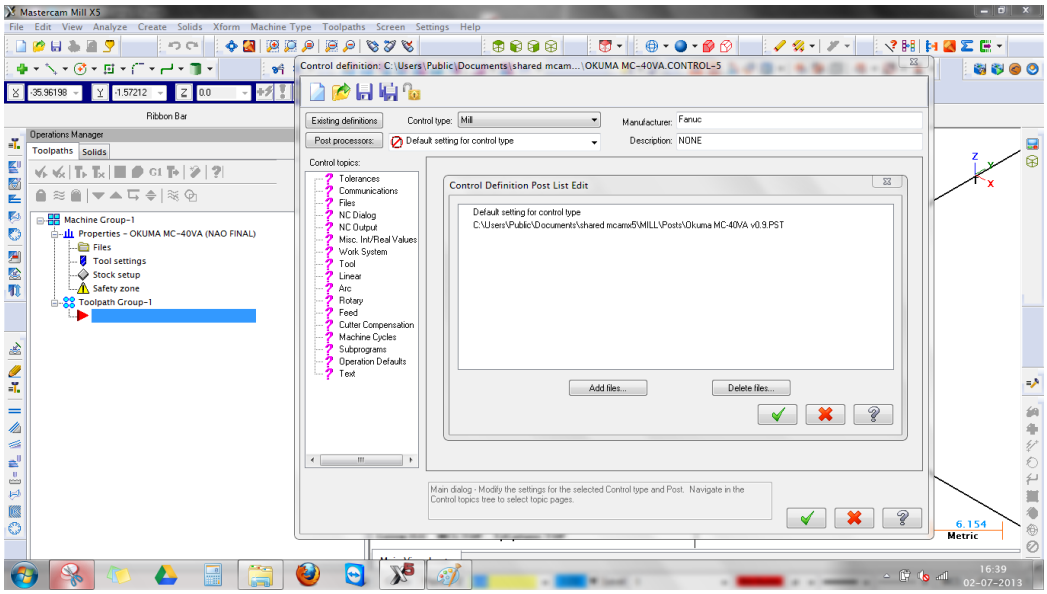


Figura 99 - Associação de um pós-processador ao ficheiro de controlo.

Gravadas as alterações ao ficheiro de controlo, e saindo desse menu, é possível agora atualizar a máquina presente na árvore de operações. Abrindo o menu da mesma, verifica-se que todas as configurações estão de acordo com o esperado (Figura 100).

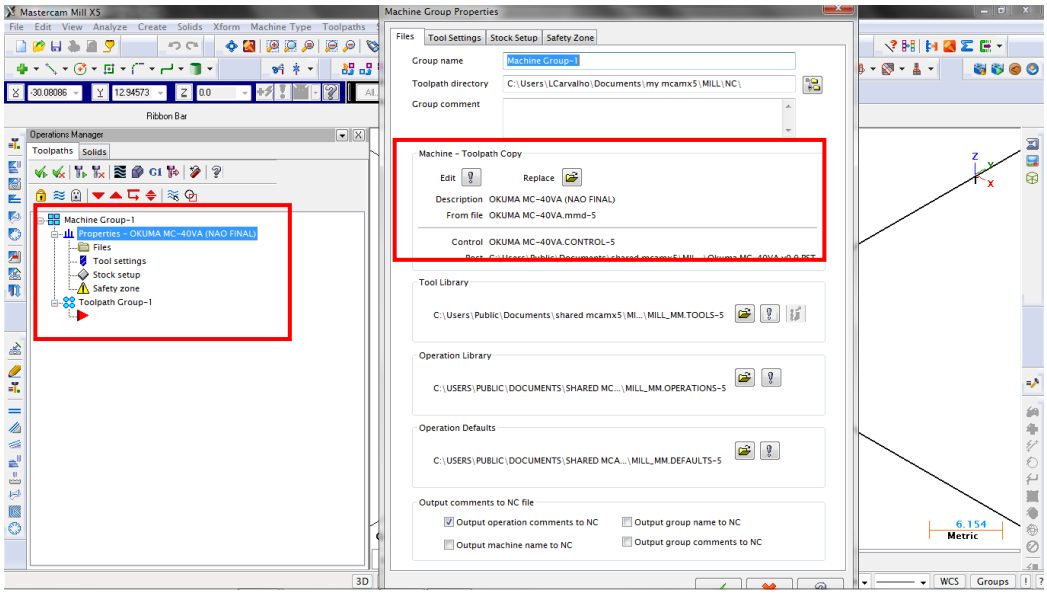


Figura 100 - Verificação das escolhas de ficheiro de controlo e pós-processador no menu da máquina virtual.

Finalmente, para ilustrar as capacidades de simulação do *software* para efeitos de deteção de possíveis colisões, apresenta-se, na Figura 101, a definição do nível de entidades geométricas no qual estará inserido o modelo da máquina virtual. Este nível deve ser um nível vazio, pois agora, o ficheiro da máquina virtual (OKUMA MC-VA40.mmd-5) terá sempre agregado o seu modelo 3D. A partir deste momento, sempre que a máquina for escolhida (Figura 85) o ficheiro *.mcx-5 passará a ter também esses modelos na sua árvore de modelação, como é possível observar na Figura 102, à esquerda.

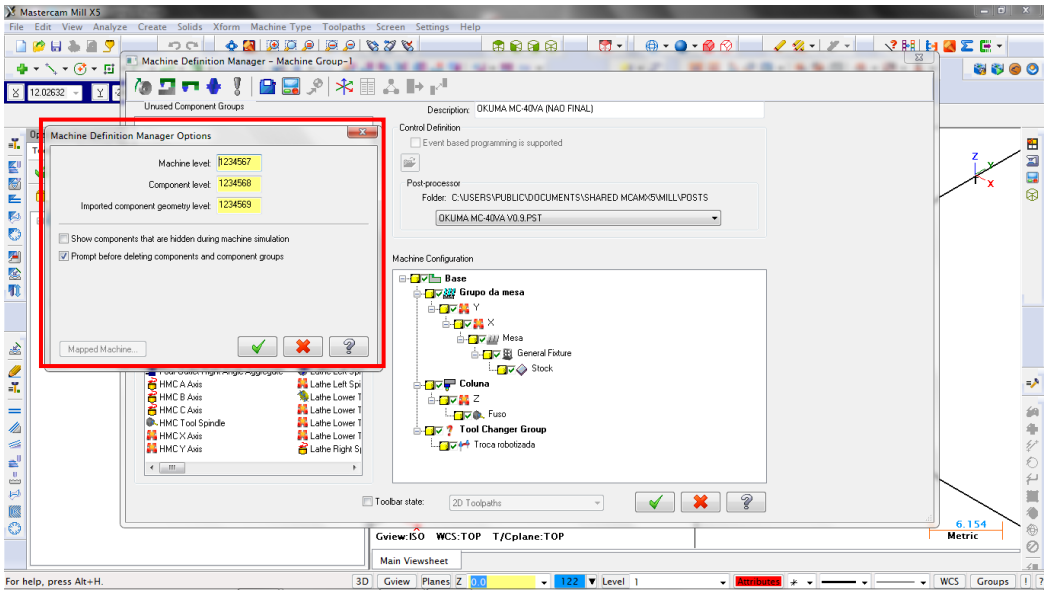


Figura 101 – Menu dos níveis onde se encontra o modelo 3D da máquina virtual.

Como se pode ver na Figura 102, já com o Modelo 1 usado nos exemplos do capítulo 4, a máquina virtual contempla, para efeitos de apresentação deste trabalho, a emulação de um sistema de fixação de peça, solidário com a mesa. A vantagem de se gerar percursos de maquinagem com o modelo da máquina virtual presente é o de detetar precocemente situações de colisão (não com deteção automática, mas sim visualizando).

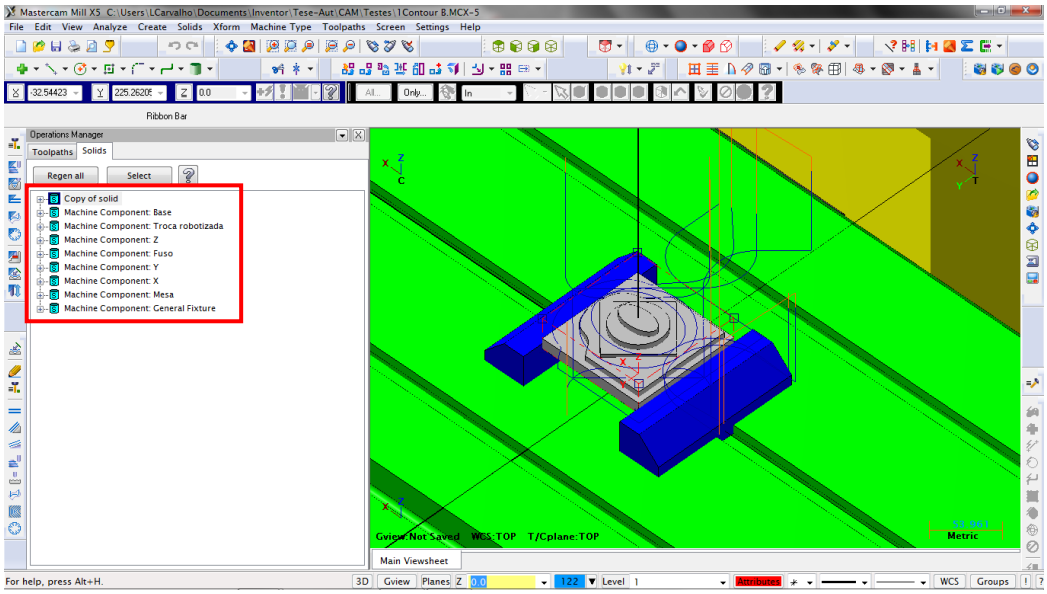


Figura 102 - Emulação de um sistema de fixação de peça.

Na Figura 103 mostra-se uma vista mais global do modelo 3D da máquina virtual, com o sistema de eixos da modelo *.mcx-5 coincidente com o zero dos eixos da máquina.

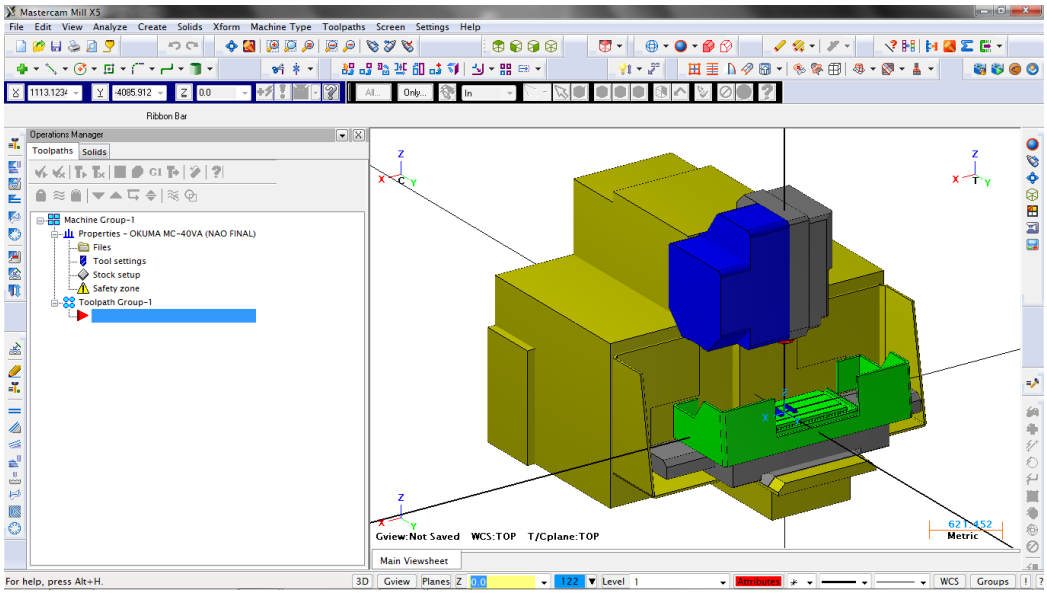


Figura 103 - Modelo 3D da máquina virtual na área de trabalho.

Após a ilustração do modelo 3D da máquina no ambiente de programação de percursos, passe-se ao ambiente de simulação. O modelo 3D da máquina de simulação não é obtido por conversão direta do modelo usado para a máquina virtual. Este deve ser inserido por outra via, cuja exemplificação não fará parte desta sequência. No entanto, esse processo é relativamente facilitado pelo facto de ambas partilharem o formato do ficheiro CAD usado (*.STL). Na Figura 104 mostra-se esse modelo em ambiente de simulação.

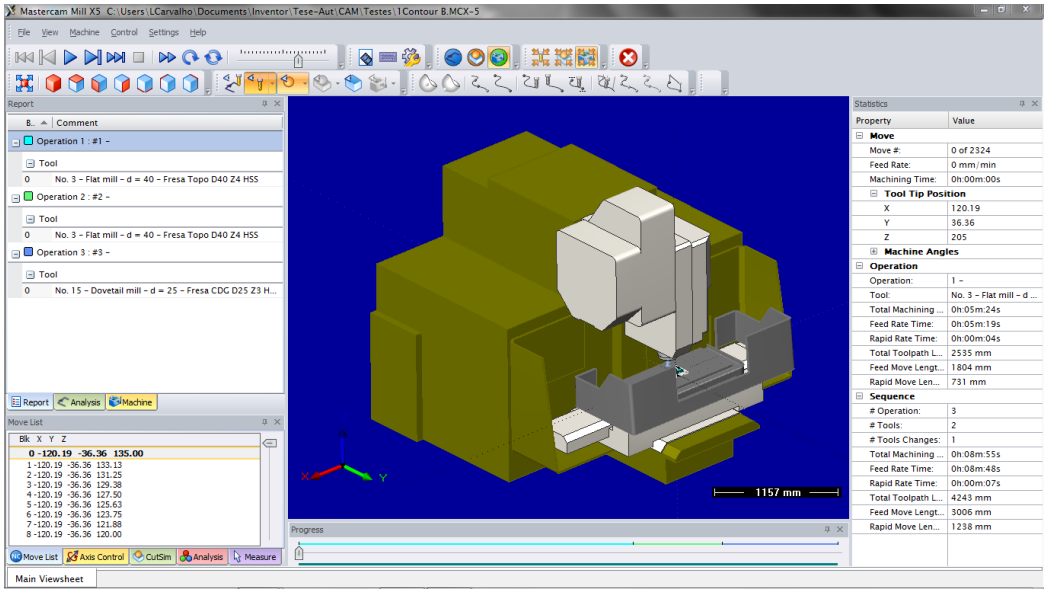


Figura 104 - Modelo 3D da máquina virtual no ambiente de simulação MachineWorks.

A simulação de operações de maquinagem, no Mastercam X5, é feita com recurso a um *software* conhecido como MachineWorks (MW), que é também utilizado por outros programas da concorrência, como o InventorCAM, por exemplo. Este *software* (MW) utiliza o percurso gerado para detetar colisões de ferramenta com outros componentes, colisões de elementos de fixação com material a cortar ou outros componentes, corte de material excessivo, e ultrapassagem dos limites. Na Figura 105 ilustra-se uma situação em que, usando o exemplo do Modelo 1, o eixo Z é obrigado a ultrapassar o seu limite inferior, pois a ferramenta é demasiado curta. Na Figura 105 é possível observar a indicação de erro por parte do *software* MachineWorks (indicado por pequenas cruzes a vermelho).

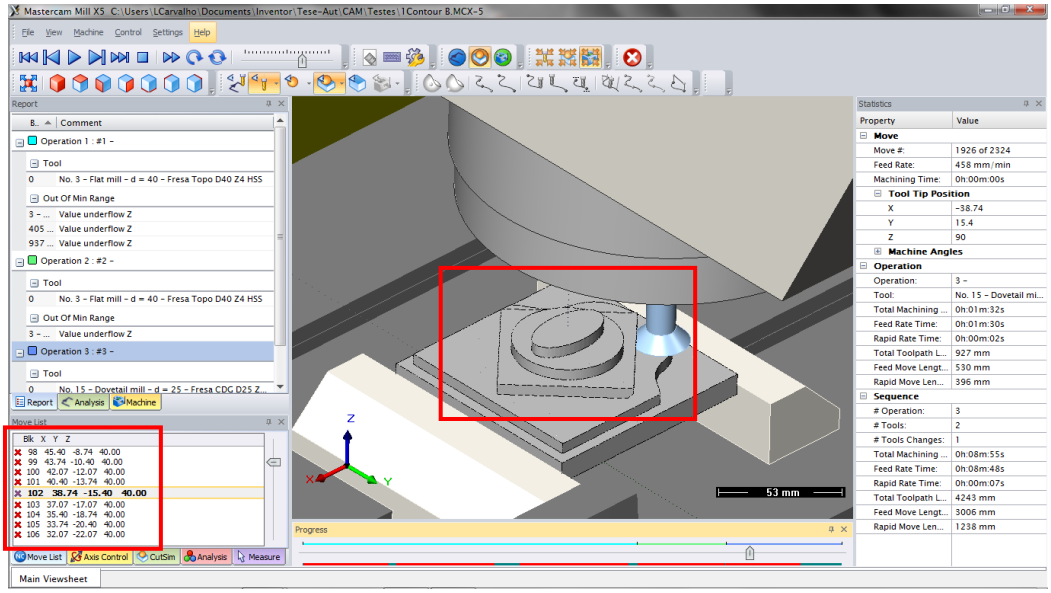


Figura 105 - Visualização da simulação de corte e controlo de colisão com o programa MachineWorks.

A partir deste ponto, o utilizador poderá:

- Gerar percursos de maquinagem observando imediatamente o seu enquadramento na máquina;
- Limitar os percursos de maquinagem aos limites de velocidade dos eixos e *spindle* da máquina;
- Estimar tempos de ciclo com velocidades mais realistas, contando também com tempos de troca de ferramenta;
- Usufruir de opções de pós-processamento especializadas para a máquina;
- Gerar código de CN que requererá poucas ou nulas edições.

Terminado este subsecção, foi possível:

- Reconhecer o significativo aproveitamento do conjunto de informações, recolhido a partir de manuais de equipamentos, na configuração de uma máquina virtual dentro do Mastercam X5, representativa de uma máquina real;
- Apreciar também as funcionalidades envolvidas neste processo assim como a estrutura de *software* que lhes está inerente, como foi explanado no capítulo 3, avaliando ainda a sua utilidade como um apoio ao planeamento tecnológico e tomada de decisão.

Este subsecção tratou, de uma forma compreensiva, dos passos necessários para a (quase) total configuração da máquina virtual, ou seja, da adaptação do Mastercam X5 para a máquina Okuma MC-40VA e controlador OSP5020M.

A importância deste subsecção reside, essencialmente, no facto de que com ele é possível transmitir uma noção clara dos requisitos de configuração de um *software* CAM em geral, e em particular para a oferta comercial que é o foco deste estudo. Sem um claro entendimento desses requisitos, não será possível realizar este exemplo de configuração de uma forma expedita, sistemática, e de elevado grau de confiança.

Essa configuração, tendo sida realizada num trabalho de nível académico, sugere que, profissionalmente, será exigido uma *performance* mais completa e mais eficiente. Posto isto, e ainda no esforço de completar esse trabalho, passa-se ao subsecção seguinte, onde serão expostas e explicadas as alterações ao pós-processador.

5.5 – Edição de pós-processadores

De todas as atividades desenvolvidas e analisadas nesta dissertação, a edição de pós-processadores é, aquela que, se extrapolada para situações reais, permite a geração de código de CN com um grau de confiança satisfatório.

Finalizando o capítulo 5, este subsecção fará a exposição do trabalho de edição realizado no pós-processador ao qual corresponde o ficheiro **Generic Fanuc 3X Mill.pst**. Como se pode adivinhar pelo seu nome, este pós-processador foi concebido para um controlador FANUC genérico, para máquinas de fresar de comando numérico. Este pós-processador foi já introduzido neste trabalho, ao ser usado para pós-processar os percursos de maquinagem produzidos no exemplo dos Modelo 1. Desta forma, como suporte a este subsecção, os resultados da avaliação do código de CN do Modelo 1 serão comparados com o código que, para o mesmo Modelo 1, se obterá usando este novo pós-processador – OKUMA MC-40VA v0.9.pst.

A adaptação do pós-processador ao controlador OSP5020M exige, de acordo com o que foi apresentado anteriormente, conhecimentos de programação de CN, entendimento da linguagem usada pela aplicação de pós-processamento (MP.dll), e o conhecimento dos códigos usados para designar dados no ficheiro *.NCI, também como foi abordado anteriormente.

A importância desta subsecção reside na exemplificação do tipo de blocos de programação que se podem implementar num pós-processador, evidenciando não só a sua flexibilidade, mas também o enorme potencial na área da automação.

A apresentação da edição do pós-processador está dividida parcialmente de acordo com o esquema apresentado na Tabela 1 (pp. 82):

1. Inicialização das variáveis e *strings* implementadas – aqui estão as introduções das novas variáveis e aplicações de formato;
1. Alteração aos blocos de início de ficheiro – aqui estão os blocos editados e adicionados para fornecer mais informação no cabeçalho do programa de CN;
2. Alteração aos blocos de pós-processamento geral – aqui estão as alterações aos blocos que estipulam o que será escrito no ficheiro de código de CN;
3. Alteração aos blocos auxiliares/implementação de novos – aqui estão as alterações e adições aos blocos que auxiliam o pós-processador, ou conferem outras funções.

Neste caso prático, salientando que se tratou da adaptação de um *software* do ano de 2010 para uma máquina que se data da década de 1990, torna-se evidente a capacidade de adaptação deste produto, no sentido em que mesmo com 20 anos de intervalo tecnológico, foi possível – naturalmente apenas até certo ponto – tirar partido de um centro de maquinagem. Apenas se pode especular quanto ao verdadeiro potencial deste tipo de ofertas, quando associadas a máquinas igualmente modernas.

Para efeitos de melhor visualização do pós-processador, as páginas onde o mesmo se encontra serão apresentadas na horizontal. Por esse motivo, e porque a se trata de matéria sujeita a uma análise detalhada que interrompe a leitura desta dissertação, o pós-processador encontra-se em dois anexos. No Anexo C, apresenta-se em excertos devidamente comentados, e no Anexo D, apresenta-se na íntegra.

6 – Avaliação da utilização do Mastercam

Atingindo este ponto da dissertação, foi atingido um nível de utilização que permite avaliar dois aspetos que se consideram importantes: a qualidade das funcionalidades do Mastercam, e a eficiência do pós-processador construído. Considerando as diversas fases ao longo deste trabalho, destacam-se alguns pontos que serão importantes no enquadramento deste capítulo:

- A abordagem do estado-da-arte do CAM no contexto global, assim como no de um *software* em particular – Mastercam X5. Isto será útil no estabelecimento de um ponto de comparação entre as funcionalidades presentes neste *software*, e outras que se encontram noutras ofertas comerciais;
- O estudo das funcionalidades desse programa e a forma como estão estruturadas. Aqui a utilidade consiste em ajudar a compreender o porquê do modo de funcionamento das funcionalidades do Mastercam;
- A exemplificação do uso do Mastercam. A experiência obtida ao longo deste trabalho foi o fator determinante na formação da opinião que se expõe neste capítulo.

Posto isto, a análise que se apresentam é dividida em dois pontos distintos:

- Avaliação das funcionalidades disponíveis no Mastercam X5;
- Avaliação do código de CN gerado pelo pós-processador alterado.

O primeiro ponto consiste na classificação compreensiva e individual de cada uma das funcionalidades antes introduzidas no capítulo 3. Aqui será feita uma análise mais descriminada das mesmas, usando um sistema de classificação que será então explicado.

O segundo ponto contrapõe o código de CN gerado na exemplo do Modelo 1, com um código de CN gerado através do novo pós-processador, apontando as melhorias resultantes das diversas alterações nele feitas.

Nas subsecções 6.3 e 6.4 faz-se um comentário às avaliações expostas.

6.1 – Avaliação das funcionalidades do Mastercam X5

Apresenta-se, nas tabelas 11 à 16, a avaliação das funcionalidades apresentadas a partir da página 25, assim como uma contraposição com o que se entende como um *software* CAM com funcionalidades avançadas ideais.

As funcionalidades avançadas ideais não são funcionalidades que representam o mais avançado do estado-da-arte (seria necessário um estudo mais pormenorizado da globalidade dos sistemas CAM), mas sim que representam o aperfeiçoamento do que já se encontra no Mastercam X5, tendendo para um enquadramento tecnológico superior. O que se descreve com estes termos é a correção dos aspetos negativos apresentados e o enriquecimento da funcionalidade através de algumas evoluções da mesma.

A avaliação de funcionalidades foca os pontos A a G anteriormente descritos – assim como as particularidades desses pontos – e o seu resultado está dividido em 4 grupos de informação:

- Importância da sua implementação – escala de 1 a 5 valores;
- Classificação da sua implementação – escala de 1 a 5 valores;
- Aspetos positivos;
- Aspetos negativos;

A escala de 1 a 5 valores, apesar de numericamente comparável, assenta num critério qualitativo, com significado facilmente apreciável, e que diz respeito à funcionalidade em si.

No grupo "Importância da sua implementação", assume-se o critério:

- ★ Descartável – a sua não existência não penalizaria o trabalho
- ★★ Acessório – de utilidade questionável
- ★★★ Útil – de utilidade comprovada
- ★★★★ Proveitoso – que adiciona valor ao *software*
- ★★★★★ Imprescindível – que é essencial

No grupo "Classificação da sua implementação", assume-se o critério:

- ★ Com falhas que comprometem a confiança no resultado final
- ★★ Com falhas que dificultam significativamente a sua utilização
- ★★★ Com falhas que tornam a sua utilização morosa
- ★★★★ Com falhas apenas a nível da sua utilidade
- ★★★★★ Com falhas pouco relevantes e desprezáveis

Este critério pretende, de uma forma abrangente, e através de uma classificação uniforme, aferir o valor de uma funcionalidade e a qualidade da sua inclusão no Mastercam X5.

Desta forma, uma funcionalidade poderá atingir nota máxima a nível da sua utilidade, mas estar mal implementada (correções enriquecerão o *software*), e por outro lado, a excelente implementação de uma funcionalidade pouco útil não lhe garantirá nota máxima, já que, subjetivamente, o seu impacto é reduzido.

A avaliação por aspetos positivos e aspetos negativos foca, essencialmente, o que faz a funcionalidade acrescentar valor ou particularidades que lhe conferem maior utilidade – no caso dos aspetos positivos – ou, no critério antónimo, o que deve ser corrigido para que a mesma seja relevante, ou para que proporcione uma experiência melhor para o utilizador.

Diferentemente do que é apresentado na Figura 4 e na Figura 5 (na subsecção 2.6), onde são apresentados conceitos tecnológicos, estas tabelas particularizam essas implementações no contexto da interface do *software* e da sua utilização[54] para o Mastercam X5.

Relembrem-se, para melhor compreensão dos mesmas, os pontos A a G introduzidos na subsecção 3.2:

A. Gestão de entidades geométricas: respeitante à criação, edição e restante manipulação de todas as entidades geométricas.
B. Funcionalidades CAM: relativo às máquinas-tipo às quais diz respeito esta aplicação de CAM em particular.
C. Gestão de máquinas virtuais: contém todas as funcionalidades associadas ao trabalho com máquinas virtuais: criação, edição e parametrização;
D. Gestão de ferramentas virtuais: diz respeito à criação, edição e manipulação das ferramentas virtuais;
E. Calculador de estratégias de fresagem: é a parte do programa responsável pelo cálculo e "dinamização[55]" dos percursos de fresagem;
F. Gestão de operações: toda a componente de criação, edição, importação, standardização, entre outros.
G. Estruturação de dados: respeitante à organização das bases-de-dados de todo programa.

Apesar de serem aqui apresentados resultados relativos à qualidade das funcionalidades do Mastercam X5, remetem-se os comentários para o subsecção 6.3.

[54] Apesar do esforço de apresentar todas as funcionalidades, note-se que ficarão sempre algumas menos relevantes ou de utilização altamente circunstancial, e que não farão parte das tabelas.

[55] Usa-se o termo "dinamização" para dar enfase a este ponto como o que faz o programa "avançar".

Tabela 11 - Tabela de funcionalidades | parte 1 | Interface; Gestão de entidades geométricas.

#	Funcionalidades	Importância	Qualidade	Aspectos positivos	Aspectos negativos	Situação ideal
0	Interface:	★★★★★	★★	Árvore de operações simples e eficaz Acesso ao "help" muito expedito Interface customizável de acordo com máquina virtual	Interface pouco intuitiva Algumas falhas a nível da sua organização Ícones pouco representativos Diversas falhas a nível gráfico (falhas na GUI)	Distribuição de comandos por *tabs*; organização de acordo com a função e grau de utilização; ícones com menos detalhe mas melhor associados à sua função
A	Gestão de entidades geométricas (designadamente CAD):					Componente de criação geométrica totalmente parametrizável com existência de gestor de parâmetros; geometrias descritivas organizadas por sketches; dimensionamento simplificado com associatividade entre dimensão e entidade geométrica; criação de features associada às operações de maquinagem; *workflow* de criação geométrica inequívoco, simplificado e gradualmente complexo conforme necessidade
1	Criação de geometrias definitivas simples e complexas	★★★★	★★★	Funcionalidade completa e bem estruturada	Sem parametrização; dimensionamento não associativo	
2	Criação de superfícies simples e complexas	★★★	★★★	Funcionalidade completa e bem estruturada	Função básica (consequência da funcionalidade anterior)	
3	Criação de sólidos a partir de geometrias definitivas	★★★	★★	Permite criar formas básicas	Faltam algumas opções; parametrização básica	
4	Criação de sólidos a partir de formas primitivas	★	★★★★	Permite criar formas básicas	Parametrização básica;	
5	Criação de *Features*	★★	★★★	É organizado na árvore do modelo	Incompleto; parametrização básica;	
6	Intermutabilidade sólidos-superfícies	★★★★	★★★★★	Funcionalidade completa		
7	Replicação e combinação de superfícies e sólidos	★★★	★★★	Funcionalidade completa; permite *nesting*	Sem parametrização; sem padronização customizável	
8	Importação de geometrias (formatos neutros e proprietários)	★★★★★	★★★★★	Funcionalidade completa; verifica qualidade do modelo	Não detecta *features*	
9	Detecção de formas parametrizáveis (limitada a formas cilíndricas)	★	★★★	Detecta formas cilíndricas e *fillets*	Limitada a apenas duas formas	
10	Exportação de geometrias	★★	★★★★★		Utilidade trivial	

Tabela 12 - Tabela de funcionalidades | parte 2 | Funcionalidades CAM; Gestão de máquinas virtuais

#	Funcionalidades	Importância	Qualidade	Aspetos positivos	Aspetos negativos	Situação ideal
B	Funcionalidades CAM:					Gama de módulo de acordo com as necessidades atuais da indústria
1	Fresadoras	★★★★★	★★★	Utilização de módulos para cada tipo de produto Adaptação automática da interface para cada módulo		
2	Tornos mecânicos					
3	Fresadoras 2,5D (*routing)*					
4	Máquinas de corte por fio de descarga elétrica (EDM)					
C	Gestão de máquinas virtuais:					Definição de máquinas virtuais incluída apenas numa aplicação externa; Todas as configurações da máquina contidas apenas na árvore de componentes; Máxima utilização de todos os parâmetros configuráveis; Criação, edição e estrutura de pós-processadores totalmente documentados; Configuração das opções do pós-processador com suporte por parte do *software*
1	Criação e configuração de máquinas	★★★★★	★★★	Configuração relativamente completa	Falhas a nível a utilização das informações definidas	
2	Modelação *in situ* dos componentes da máquina	★★★★	★★★★★	Articulado com a gestão de entidades geométricas	Implementação confusa e pouco intuitiva	
3	Adição de componentes em formato neutro	★★★	★★	Permite adicionar geometria à máquina virtual	Utilidade reduzida; adiciona pouco valor ao produto	
4	Adição de componentes em formato Mastercam X5	★★	★★		Implementação confusa e pouco intuitiva	
5	Assemblagem e representação local dos componentes da máquina	★★★	★★★	Funcionalidade totalmente flexível	Utilização pouco intuitiva;	
6	Definição de parâmetros associados à cinemática da máquina	★★★★	★★★★	Organização simples e eficaz	Fraca utilização das informações fornecidas pelo utilizador	
7	Definição e emulação do controlador da máquina	★★★★★	★★★★	Permite configurar aspetos básicos do controlador	Acesso a configurações dificultada por interface mal concebida	
8	Criação e edição de pós-processadores	★★★★★	★★		Pouco suporte por parte do *software*; mal documentado	

Tabela 13 - Tabela de funcionalidades | parte 3 | Gestão de ferramentas virtuais

#	Funcionalidades	Importância	Qualidade	Aspectos positivos	Aspectos negativos	Situação ideal
D	Gestão de ferramentas virtuais:					Gestão de ferramentas em aplicação externa; inclusão de gestor de elementos de fixação (cones); inclusão de fresas de pastilhas e respectivas pastilhas de corte; criação de perfis específicos com definição de caminhos de maquinagem específicos; formulador da relação material ferramenta - material a maquinar com leitura de propriedades de dureza e resistência mecânica do material
1	Criação e definição de ferramentas de formatos típicos	★★★★★	★★★★★	Funcionalidade completa; simples e eficaz	Não inclui pastilhas de corte e respectivos suportes	
2	Criação e definição de ferramentas de formatos especiais	★★★	★★★★	Funcionalidade flexível	Utilização pouco intuitiva; requer aprendizagem mais demorada	
3	Parametrização de ferramentas	★★★★★	★★★★	Funcionalidade completa; simples e eficaz	Não inclui gestão de cones de fixação neste ponto	
4	Interoperabilidade de ferramentas com definições de materiais	★★★★★	★★★	Funcionalidade simples	Formulação de velocidades não existe (é caso-a-caso)	
5	Interoperabilidade de ferramentas com estratégias de maquinagem	★★★★	★★★	Sugestão de ferramenta de acordo com a operação	Sugestão não é suportada; exige confiança por parte do utilizador;	
6	Importação de ferramentas a partir de ficheiros de texto	★★	★★★★★		Pouco útil	
7	Exportação de ferramentas para ficheiros de texto	★★	★★★★★		Pouco útil	
8	Geração de relatórios de listagem de equipamento ferramental	★★★★	★★★★★	Permite criar listas detalhadas com representação 3D		

Tabela 14 - Tabela de funcionalidades | parte 4 | Motor de cálculo de estratégias de fresagem.

#	Funcionalidades	Importância	Qualidade	Aspectos positivos	Aspetos negativos	Situação ideal
E	Motor de cálculo de estratégias de fresagem:					Geração de caminhos com pré-deteção de colisão; Utilização das informações dos limites dos eixos de movimento da máquina; *Teaching* de caminhos de maquinagem; Articulação automática com informação metrológica obtida posteriormente em controlo de qualidade; Interface inequívoca e *user-friendly*
1	Para furação, mandrilagem e rosqueamento	★★★	★★★★	Funcionalidade completa	Falhas menores na seleção de geometrias;	
2	2D simples, de contorno, abertura de cavidades e facejamento	★★★★★	★★★★	Funcionalidade completa	Seleção de geometrias pouco intuitiva; mal organizada	
3	2D a alta velocidade	★★★★	★★★★	Funcionalidade completa	Seleção de geometrias pouco intuitiva;	
4	Fresagem por penetração	★★★★	★★★★	Funcionalidade completa	Seleção de geometrias pouco intuitiva;	
5	3D de desbaste e acabamento	★★★★★	★★★★	Funcionalidade completa	Seleção de geometrias pouco intuitiva;	
6	3D de ampla gama de percursos	★★★★	★★★★	Funcionalidade completa	Seleção de geometrias pouco intuitiva;	
7	3D otimizadas para formas específicas	★★★★★	★★★★	Funcionalidade completa	Seleção de geometrias pouco intuitiva;	
8	3D otimizadas para alta velocidade	★★★★	★★★★	Funcionalidade completa	Seleção de geometrias pouco intuitiva;	
9	3D com funcionamento híbrido (estratégias combinadas)	★★★★	★★★★	(implícito nos pontos anteriores)	(implícito nos pontos anteriores)	
10	3D com 5 eixos	★★★★★	★★★	Funcionalidade completa	Processo altamente complicado; requer aprendizagem demorada	
11	Geração automática de estratégias de furação	★★	★★★	Funcionalidade configurável	Exige edição posterior; utilidade reduzida;	
12	Geração automática de estratégias de fresagem	★★	★★	Funcionalidade configurável	Exige edição posterior; utilidade reduzida; Configuração insuficiente	

Tabela 15 - Tabela de funcionalidades | parte 5 | Gestão de operações.

#	Funcionalidades	Importância	Qualidade	Aspetos positivos	Aspetos negativos	Situação ideal
F	Gestão de operações:					Articulação total com o modelo da máquina virtual (inclusive otimização de posicionamento da peça a maquinar para redução máxima de tempos de ciclo); Simulação e cálculo de tempos de ciclo totalmente integrado no *software* base; Criação de operações-tipo para ferramentas específicas de formato customizável
1	Criação e parametrização de operações	★★★★★	★★★★	Funcionalidade altamente configurável	(implícito nos pontos anteriores - E)	
2	Interoperabilidade de ferramentas e operações	★★★★★	★★★	Grande utilidade	Sugestão não é suportada; exige revisão posterior	
3	Importação de parâmetros de operações	★★★	★★★★★	Permite partilhar parâmetros entre modelos diferentes		
4	Gestão de sistemas de coordenadas	★★★★★	★★	Permite definir sistemas de coordenadas	Pouco intuitivo; requer revisão cuidada; processo confuso	
5	Exportação de operações definíveis como standard	★★★★	★★★★★	Funcionalidade completa		
6	Simulação de operações em ambiente 3D	★★★★	★★★★	Permite estimar a qualidade da operação	Não utiliza modelo da máquina virtual	
7	Cálculo de colisão	★★★★★	★★★★	Grande utilidade	Requer configuração demorada; requer revisão dos modelos CAD	
8	Cálculo de tempos de ciclo	★★★★★	★★★★★	Grande utilidade; facilmente configurável		

Tabela 16 - Tabela de funcionalidades | parte 6 | Estruturação de dados

#	Funcionalidades	Importância	Qualidade	Aspetos positivos	Aspetos negativos	Situação ideal
G	Estruturação de dados:					Gestão de todas as base de dados em aplicação externa; Inclusão de gestor de projetos com funcionalidade total e utilização completa de base de dados e inclusão em aplicação externa;
1	Gestão automática de pastas de projeto	★★	★★★★		Simplicidade reduz a sua utilidade	
2	Gestão de base de dados de definições de máquinas	★★★★	★★	Permite gerir todas máquinas virtuais	Pouco intuitivo; utilização confusa; requer aprendizagem demorada	
3	Gestão de base de dados de componentes de máquinas	★★	★★	Permite gerir os componentes das máquinas	Gestor complicado; utilidade muito reduzida	
4	Gestão de base de dados de ferramentas	★★★★★	★★★	Grande utilidade	Pouco intuitivo; utilização confusa;	
5	Gestão de base de dados de ficheiros de controlo (pós-processamento)	★★★	★★	Permite gerir os ficheiros de controlo e pós-processadores	Pouco intuitivo; utilização confusa; requer cuidado constante	
6	Gestão de base de dados de operações	★★★	★★★★	Funcionalidade completa	Utilização confusa	

6.2 – Avaliação do código de CN gerado

O código de CN gerado com recurso a uma máquina virtual não configurada e a um pós-processador pré-definido (i.e., genérico), em virtualmente todas as situações, requererá posterior edição. Esta, caracterizada pelo seu consumo de tempo e possibilidade de erro humano, sugere que outras formas automatizadas de geração de código de CN sejam usadas. Nestes casos a tradução de percursos, e as funções modais auxiliares e especiais que a acompanham, figurarão no código de CN apenas de acordo com regras pré-definidas. Graças a essas regras – o cerne do pós-processador – essa lógica será permanente, e criará uma base de confiança que culminará na não necessidade de edição.

Essa premissa requer, numa fase prévia, o preenchimento dos requisitos apresentados na página 84, e posteriormente, uma fase de testes e aperfeiçoamento.

Tal como mencionado na apresentação deste capítulo, o resultado em si consiste na exposição de um código de CN que à saída do pós-processamento, não foi editado.

Para tornar esta comparação mais completa, alteram-se as operações do Modelo 1 para que o seu percurso inclua agora uma replicação em padrão retangular (200x200 mm), de forma a que o código de CN exiba as funções associadas. Na Figura 106 mostra-se essa replicação, assim como os limites dos eixos (linhas azuis).

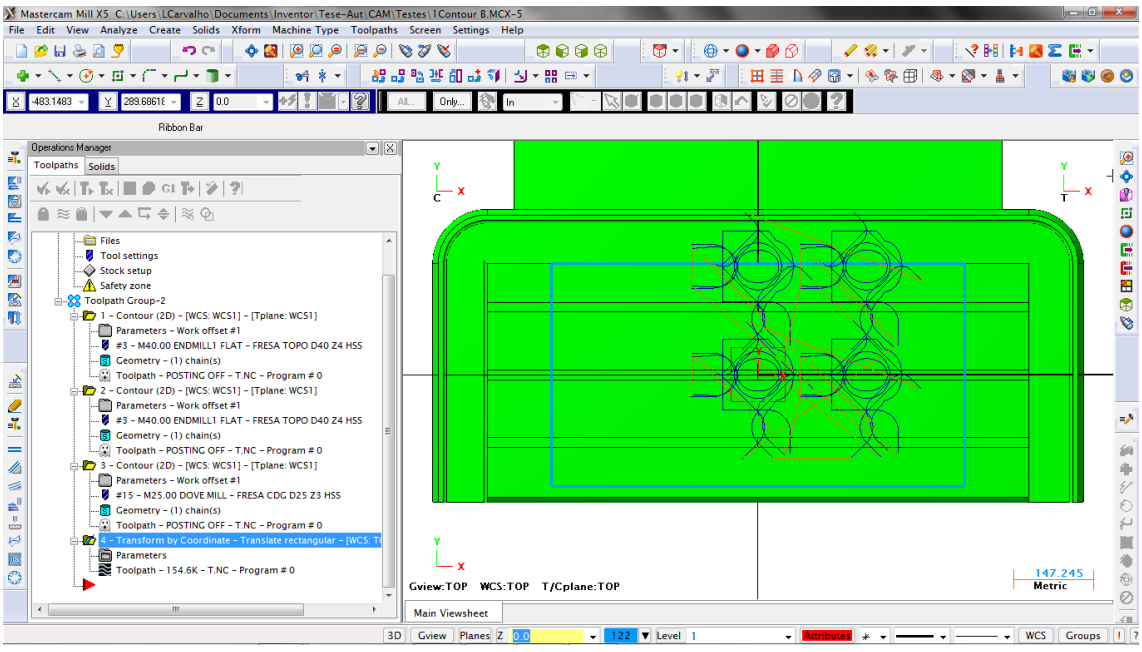

Mastercam Mill X5 C:\Users\LCarvalho\Documents\Inventor\Tese-Aut\CAM\Testes\1Contour B.MCX-5
File Edit View Analyze Create Solids Xform Machine Type Toolpaths Screen Settings Help
Ribbon Bar
Operations Manager
Toolpaths Solids
Files
Tool settings
Stock setup
Safety zone
Toolpath Group-2
1 - Contour (2D) - [WCS: WCS1] - [Tplane: WCS1]
Parameters - Work offset #1
#3 - M40.00 ENDMILL1 FLAT - FRESA TOPO D40 Z4 HSS
Geometry - (1) chain(s)
Toolpath - POSTING OFF - T.NC - Program # 0
2 - Contour (2D) - [WCS: WCS1] - [Tplane: WCS1]
Parameters - Work offset #1
#3 - M40.00 ENDMILL1 FLAT - FRESA TOPO D40 Z4 HSS
Geometry - (1) chain(s)
Toolpath - POSTING OFF - T.NC - Program # 0
3 - Contour (2D) - [WCS: WCS1] - [Tplane: WCS1]
Parameters - Work offset #1
#15 - M25.00 DOVE MILL - FRESA CDG D25 Z3 HSS
Geometry - (1) chain(s)
Toolpath - POSTING OFF - T.NC - Program # 0
4 - Transform by Coordinate - Translate rectangular - [WCS: T
Parameters
Toolpath - 154.6K - T.NC - Program # 0
Gview:TOP WCS:TOP T/Cplane:TOP
147.245
Metric
Main Viewsheet
3D Gview Planes Z 0.0 122 Level 1 Attributes WCS Groups

Figura 106 - Replicação das 3 operações associadas ao Modelo 1, e mesa de trabalho.

De seguida, apresentam-se excertos do código #1, gerado pelo pós-processador genérico, comentado onde se entende relevante:

```
%  (O início do programa de CN, não seria reconhecido pelo OSP5020M.)
O0000(MODEL1OLD)
(DATE=DD-MM-YY - 26-06-13 TIME=HH:MM - 20:55)
(MCX FILE - C:\USERS\...\MODELO1.MCX-5)
(NC FILE - C:\USERS\...\MODEL1OLD .NC)
(MATERIAL - ALUMINUM MM - 2024)
( T3 | FRESA TOPO D40 Z4 HSS | H3 )
( T15 | FRESA CDG D25 Z3 HSS | H15 )
(As linhas acima estão de acordo com o que se pretende.)
N100 G21
N102 G0 G17 G40 G49 G80 G90
(O código G49 não está de acordo com o OSP5020M.)
N104 T3 M6
N106 G0 G90 G55 X-70.189 Y13.644 S954 M3
(O código G55 não está de acordo com o OSP5020M, e falta o registo de memória.)
N108 G43 H3 Z80.
(O código G43 não está de acordo com o OSP5020M.)
N110 M98 P0001  (Operação 1)
N112 G90 X129.811 Y13.644
(Os códigos M98 e P0001 não estão de acordo com o OSP5020M.)
(...)
N126 M98 P0002  (Operação 2)
N128 G90 X221.716
(Aqui chama-se a sub-rotina #2.)
(...)
N140 M5
N142 G91 G28 Z0.
N144 M01
N146 T15 M6
(Aqui dá-se a mudança de ferramenta – entra aqui a fresa cauda-de-andorinha.)
N148 G0 G90 G55 X150.104 Y25.251 S1527 M3
N150 G43 H15 Z80. (Esta linha apresenta um erro que será explicado adiante.)
N152 M98 P0003  (Operação 3)
N154 G90 X350.104 Y25.251
(...)
N166 M5
N168 G91 G28 Z0.  (Aqui coloca-se a máquina na posição CASA. (G28).)
N170 G28 X0. Y0.
N172 M30

O0001
N100 G91
(As posições ao longo das sub-rotinas são programadas relativamente (G91), pois da outra forma, no modo absoluto (G90), usar-se-iam N registos de sistemas de eixos, o que requer programação prévia no controlador.)
N102 Z-15.
(...)
N370 G0 Z100.
N372 M99  (O retorno da sub-rotina é feito com M99, em vez de ser com RTS.)
%
```

Em linha com os comentários feitos na página anterior, continue-se a analisar este código, referindo alguns aspetos:

- A máquina Okuma tem os seguintes limites de curso dos eixos (mm):
  - Limite máximo em X: 381; Limite mínimo em X: -381;
  - Limite máximo em Y: 205; Limite mínimo em Y: -205;
  - Limite máximo em Z: 650; Limite mínimo em Z: 200.
- As ferramentas T3 e T15 têm os seguintes comprimentos (mm):
  - T3 : H3 = 70;
  - T15 : H15 = 50.
- O registo do sistema de eixos **G55** corresponde à posição (-50,-50, 55).

De acordo com estes dados, e observando a linha N150:

- `N150 G43 H15 Z80.`

Constata-se que em N150, a ferramenta está ser enviada para uma posição que a máquina não alcança: limit_z_min - H15 - G55z = 200-50-55 = 95 > 80. Ou seja, a posição mínima que esta ferramenta pode atingir, em Z, é 95 mm. Posto isto, este código de CN não iria produzir o efeito esperado.

Analisados estes excertos, faça-se agora a análise dos excertos correspondentes do código #2 gerado com o pós-processador modificado:

```
$MODELO1NEW.MIN%
(Este início já está de acordo com o OSP5020M.)
O0000(MODELO1NEW)
(DATE=DD-MM-YY - 26-06-13 TIME=HH:MM - 20:56)
(MCX FILE - C:\USERS\...\MODELO1.MCX-5)
(NC FILE - C:\USERS\...\MODELO1NEW.NC)
(MATERIAL - ALUMINUM MM - 2024)
(Até este ponto o cabeçalho mantém-se idêntico.)
(LIMITES - CODIGOS DE ERRO:)
(XFDL - X FORA DE LIMITES)
(YFDL - Y FORA DE LIMITES)
(ZFDL - Z FORA DE LIMITES)
(LIMITES DOS EIXOS :)
(X [-381;381])
(Y [-205;205])
(Z [200;650])
(Aqui como se pode ver estão as modificações ao cabeçalho, com a inserção dos
códigos de erro, e os limites de curso dos eixos.)
(T3|FRESA TOPO D40 Z4 HSS|H3)
(T15|FRESA CDG D25 Z3 HSS|H15)
N100G21
N110G0G17G40G53G80G90
(Aqui o cancelamento do offset do comprimento da ferramenta está com o código
modal correto (G53).)
```

```
N120T3M6
N130G15H01
```
(Na linha cima chama-se a posição de sistemas de eixos com o registo de memória **H01**, através da função modal **G15**. Esta chamada utiliza a informação definida na própria operação, e a sua escolha não requer qualquer intervenção manual no pós-processador.)
```
N140G0G90X-70.189Y13.644S954M3
N150G56H3Z80.
```
(O offset da ferramenta **T3** é feito aqui, com a função modal certa: **G56**, que aqui usa o registo de memória **H3**. De notar que, como a ferramenta T3 tem um comprimento superior, a posição Z80 não está fora dos limites, e portanto esta linha não é assinalada com uma mensagem de erro.)
```
N160CALL OSUB1 (Operação 1)
```
(A chamada da sub-rotina é feita na linha acima. "**CALL OSUBx**" é o formato correto, usado no OSP5020M.)
```
N170G90X129.811Y13.644
N180CALL OSUB1
N190G90X-70.189Y213.644(<YFDL)
```
(Aqui posição Y213.644 aparenta estar fora dos limites dos eixos. No entanto, este código ainda não considera saídas dos limites com a relativização ao *offset* do sistema de eixos. Neste caso em particular a posição Y213.644 não ultrapassou o limite, pois considerando **G15 H01,** a este valor deve-se deduzir 55 mm. Aqui exemplifica-se algo que deve ser melhorado no pós-processador.)
```
(...)
N240CALL OSUB2 (Operação 1)
N250G90X221.716
(...)
N310M5
N320G91G30 P0
```
(A posição CASA aqui é chamada com **G30 P0**, de acordo com o OSP5020M.)
```
N330M01
N340T15M6
N350G0G15H01
N360X150.104Y25.251S1527M3
N370G56H15Z80.(<ZFDL)
```
(A linha acima representa a mesma definição de *offset* da ferramenta através da função modal **G56**, colocando depois a ferramenta na posição Z80. No entanto, devido às alterações no pós-processador, dá-se a deteção de erro, inserindo a mensagem de erro no local: **ZFDL**.)
```
(...)
N460M5
N470G91G30 P0
N480M30
OSUB1
N100G91
N110Z65.Z-15.
(...)
N1500G0Z100.
N1510RTS (Aqui o retorno é feito com a função correta: RTS.)
(...)

%
```

Esta análise conclui a comparação dos códigos de CN produzidos através de duas vias distintas. Resumindo as diferenças que são possíveis observar por comparação dos dois códigos:

1. No código #2 o início do cabeçalho está de acordo com o controlador OSP5020M. No código #1 não está;
2. No cabeçalho do código #2 existem linhas que informam acerca dos limites de curso dos eixos e das mensagens de erro que são apresentadas quando estes são ultrapassados. No código #1 essas linhas não existem;
3. No código #2 o cancelamento da função modal de *offset* da ferramenta (G53) assim como o novo *offset* a aplicar (G56) estão de acordo com o controlador OSP5020M. No código #1 estão de acordo com um controlador FANUC genérico;
4. No código #2 a chamada da mudança de sistema de eixos é feita com a função modal G15, com atribuição de registos, de acordo com o controlador OSP5020M. No código #1é feita com G55, de acordo com um controlador FANUC genérico;
5. No código #2 a função que designa a posição CASA é G30, como especifica o controlador OSP5020M. No código #1 esta posição é chamada com G28;
6. No código #2 as chamadas e retornos de sub-rotinas são feitas com recurso a CALL OSUBx e RTS. No código #1 são feitas com recurso a M98 e M99, respetivamente.
7. O código #1 não deteta erros de ultrapassagem de eixos, deixando essa deteção para quando o código já foi inserido na máquina. No código #2 essa deteção é feita no pós-processamento.

Após esta comparação, termina-se este capítulo apresentando as funções de controlo de pós-processamento, que terminam o mesmo em caso de erro:

```
(...) (Até este ponto consiste apenas o cabeçalho.)
N100G21
N110G0G17G40G53G80G90
N120T3M6
N130G15H01
N140G0G90X-70.189Y13.644S954M3
N150G56H3Z80.
N160CALLOSUB1
N170G90X129.811Y13.644
N180CALLOSUB1
N190G90X-70.189Y213.644(<YFDL)(VERIFICAR LIMITES DO PROGRAMA :
POST TERMINADO)
(Na linha acima representada, verifica-se a interrupção do pós-processamento,
aquando da deteção de um erro de limites.)
```

```
(...) (Até este ponto consiste apenas o cabeçalho.)
N100G21
N110G0G17G40G53G80G90
N120T3M6
N130(SISTEMA DE EIXOS G15 DEVE SER : H01-H50 ou H101-H150 .....
POST TERMINADO) (Aqui interrompe-se o pós-processamento por erro em G15.)
```

Chegado a ponto da dissertação, podem-se discutir os resultados obtidos. Devido à estrutura deste trabalho, e à sua natureza, esta discussão não incidirá só sobre os resultados apresentados no capítulo anterior, mas também sobre a experiência do estudo da estrutura do Mastercam X5, no capítulo 3, e sobre as atividades realizadas no capítulo 4. Entende-se que a interação com o *software* que foi sucedendo ao longo da dissertação constitui um resultado qualitativo, e acima de tudo formador de opinião.

Isto sugere que a discussão neste capítulo se divida em 3 partes:

1. Comentário à avaliação das funcionalidades do Mastercam X5;
2. Comentário à estrutura de funcionamento do Mastercam X5;
3. Comentário à edição de pós-processadores.

A leitura destes quatro pontos pode levar à noção de que esta discussão assenta apenas numa base de subjetividade. De facto, dos resultados apresentados, só a comparação de códigos de CN é passível de ser interpretada como algo palpável e enquadrável num critério objetivo. Mas mesmo aí entende-se que a verdadeira apreciação dos resultados dessa fase passará pelo conhecimento prévio do trabalho com pós-processadores, e/ou sentimento de necessidade do seu uso, derivado da eventual correção de códigos de CN. Isto é, através do reconhecimento da sua utilidade.

Relembre-se que o código de CN segue – à exceção de algumas funções como as que foram sendo observadas ao longo do trabalho – uma norma ISO [1]. Este facto leva a que, havendo uma máquina e controlador bem estudados (neste caso, a Okuma MC-40VA e o controlador OSP5020M), e exemplos de código de CN fornecidos pelos seus fabricantes[56], o teste de códigos de CN torna-se uma questão determinística. Ou seja, devido ao próprio conceito de programação, ações iguais produzem efeitos iguais. Posto isto, este raciocínio é aplicado apenas como apoio à edição do pós-processador, considerando-se imprescindível testar os códigos por ele gerados na versão real da Okuma MC-40VA.

[56] Para este estudo, foram fornecidos, por parte da Faculdade de Engenharia da Universidade do Porto, alguns exemplos de programas de CN feitos para a Okuma MC-40VA.

6.3 – Comentário à avaliação das funcionalidades do Mastercam X5

As funcionalidades que o Mastercam X5 disponibiliza no seu *software* são, de acordo com a experiência deste trabalho, de reconhecida utilidade. Fazendo uso da perspetiva introduzida no início desta dissertação, como apoio à tomada de decisão associada ao planeamento tecnológico, o Mastercam X5 cumpre as suas funções.

O critério aplicado nas tabelas de avaliação das funcionalidades baseou-se quase exclusivamente na promoção de eficiência. Com isto quer-se dizer que com este critério, o valor das funcionalidades avalia-se usando três pontos de vista:

- É reconhecido o impacto a nível da aceleração do trabalho para o qual o Mastercam X5 está pensado (ou seja a produção de peças através de processos de maquinagem).
- É reconhecido o impacto a nível da qualidade da informação disponibilizada como apoio à tomada de decisão;
- É reconhecido impacto a nível da aceleração do próprio trabalho com o Mastercam X5.

No primeiro ponto inserem-se as funcionalidades relacionadas com a geração de código de CN específico, geração de percursos otimizados para poupança de tempo, para formas específicas, melhor utilização da ferramenta, entre outros. As funcionalidades, sobre esta perspetiva, adicionam valor ao *software* se, de uma forma significativa, causarem maior eficiência na produção. Fala-se, por exemplo, de uma funcionalidade que de facto leva a que se maquine uma mesma peça, com a mesma qualidade, mas com uma redução mensurável de tempo.

No segundo ponto estão as funcionalidades que permitem estimar o tempo de ciclo, simular a qualidade do acabamento e simular a ferramenta em ambiente máquina, detetando colisões. As funcionalidades neste ponto são valiosas se acelerarem o processo de tomada de decisão, fornecendo dados concretos de forma expedita, mas mais importantemente, fornecendo informação mais próxima da realidade.

No terceiro ponto está tudo o que se relaciona com a estrutura do *software*, interface, organização de dados e assistência à edição de pós-processadores, por exemplo. A valorização, segundo esta lógica, é realizada em medida da aceleração da utilização dos primeiros dois pontos.

Avaliar as funcionalidades de uma oferta comercial de CAM segundo este critério obriga – de certa forma – a experimentar-se o programa em questão. Só assim é possível ter uma medida real da sua qualidade.

Comentando a avaliação em si, apresentada desde a Tabela 11 até à Tabela 16, é observável os seguintes aspetos:

- O critério de importância atribuído às funcionalidades CAD requer uma prévia explicação. Do ponto de vista de *software* CAM, a importância de funcionalidades CAD (criação e deteção de *features*, exportação de sólidos, p.e.) não é valorizada. Para um *software* CAM o essencial é poder importar sólidos e converte-los para elementos mais básicos e manipulá-los, assim como criar geometrias descritivas e superfícies. Do ponto de vista do *software* CAD/CAM, as funcionalidades CAD já são mais valorizadas. Posto isto o Mastercam X5 é muito bom nessas funcionalidades CAD limitadas, e é fraco nas funcionalidades

mais avançadas. A sua interface pouco intuitiva dificulta a utilização destas funcionalidades;

- A nível de funcionalidades CAM (aqui faz-se referência às funções de CAM tipificadas de acordo com a máquina), as quatro possibilidades que a versão usada Mastercam X5 oferece, fazem todas parte de um conjunto de módulos. Esta política modular valoriza o produto globalmente, pois permite concentrar esforços financeiros apenas onde é necessário, evitando despesas desnecessárias. Associado a cada módulo está uma interface que para cada um apresenta algumas diferenças;
- A funcionalidade de gestão de máquinas virtuais é completa e permite configurar a máquina virtual com um grau de diversidade significativo. Isto fornece várias vantagens a nível de interação com o pós-processador, pois é útil definir parâmetros associados ao pós-processamento através das configurações da máquina virtual no Mastercam X5, visto que da forma contrária seria necessário editar o pós-processador diretamente. A gestão de máquinas virtuais, à semelhança do que acontece com as funcionalidades CAD, peca por uma interface confusa, e uma estrutura complicada. Várias vezes, ao longo da utilização do Mastercam X5, deram-se erros devido a estes fatores. Quanto ao apoio à edição de pós-processadores, este é dado em apenas um documento, que não detalha nem o funcionamento do pós-processador, nem o funcionamento do MP.dll, fazendo apenas referência às variáveis que estão associadas à máquina virtual, parâmetros de operações, e de ferramentas. Este nível de apoio é insuficiente.
- Relativamente à gestão de ferramentas constata-se que estas funcionalidades são comprovadamente úteis e, a par de alguns problemas relacionados com – mais uma vez – a interface, estão bem implementadas. A criação e edição de ferramentas são particularmente expeditas. Algumas dificuldades surgem na criação de ferramentas com perfil customizável. A nível da interação material ferramenta, que é tabelar, e não por formulação[57], considera-se que esta é uma área a melhorar.
- As funcionalidades de planeamento de estratégias de fresagem são as que mais influenciam a qualidade geral do Mastercam X5. Nesta oferta comercial esta área está bem desenvolvida e as operações que podem ser criadas com estas estratégias são flexíveis e parametrizáveis, e cobrem uma área de aplicação apreciável. Aqui, tal como nos outros pontos, a interface pouco intuitiva, e de utilização confusa, assim como um mecanismo de seleção de geometrias mal desenhado, levam a que se perca algum do potencial destas funcionalidades. O planeamento automático de estratégias por deteção de *features* é uma implementação que ainda requer alguma evolução. Entende-se que esta funcionalidade requer uma componente de CAD completa, algo que no Mastercam X5 não existe;
- A gestão de operações é constituída por funcionalidades que permitem criar, editar, manipular e realizar outras tarefas menos relevantes como importar ou exportar operações. Aqui incluem-se as funcionalidades de simulação de remoção de material e de colisão. Incluem-se também a interoperabilidade

[57] Aqui faz-se a distinção entre a aplicação de valores de corte mediante o material a cortar, baseada em tabelas pré-definidas, da aplicação baseada numa formulação geral.

operação-ferramenta, que necessita de evolução, e a gestão de sistemas de eixos, que igualmente requer atenção, por ser de utilização confusa. Posto isto, e à exceção destes pontos, o restante está bem implementado e tem um impacto significativo na qualidade geral do Mastercam X5 – muito devido à componente de simulação;

- Por fim, relativamente à estruturação de dados, tem-se de um lado a forma como os dados estão organizados entre si, e do outro a forma como os dados são manipulados. Aqui encontram-se diversas dificuldades, pois os ficheiros das bases-de-dados estarem distribuídos em vários locais no computador, o que requer à partida que o utilizador tenha uma utilização do seu posto de trabalho organizada por si só. Além disto, o próprio Mastercam X5 tem vários pontos de acesso às bases-de-dados, algo que, à primeira vista pode indicar flexibilidade, mas que mais tarde se releva confuso.

Termina-se este comentário com a seguinte observação: o Mastercam possui excelentes capacidades de planeamento de percursos e de implementação de máquinas virtuais. A gestão de ferramentas é simples mas eficaz. A interface pouco intuitiva e a sua estrutura, que falha por ser confusa, dificultam o aproveitamento das suas qualidades, ocultando o seu pontencial. O melhoramento destes aspetos negativos contribuirá significativamente para que o Mastercam seja um *software* de excelência.

6.4 – Comentário à edição de pós-processadores

O resultado apresentado na subsecção 6.2 é apenas uma amostra do que é possível obter-se com um pós-processador. De facto, tendo em conta a notável flexibilidade que esta estrutura de processamento oferece, não se justifica ignorar-se esse potencial.

O código gerado pelo pós-processador modificado não sofreu quaisquer edições. Como explicado nesse subsecção, além da possibilidade de tirar partido de funções do controlador da máquina às quais o *software* em questão – neste caso o Mastercam X5 – poderá até nem ter acesso (referem-se em particular as funções especiais do controlador), há ainda a possibilidade de o fazer sem edição posterior.

Garantidas condições que permitam confiar no pós-processador usado, a sua implementação é crucial no desenvolvimento de sistemas de elevada autonomia. Assim sendo, na perspetiva deste trabalho, mais precisamente na análise do resultado obtido com a edição de um pós-processador, pode-se concluir que:

- Um pós-processador promove autonomia;
- Um pós-processador contribui para a eficiência, através da redução de tempo despendido na edição de código de CN;
- Um pós-processador exige requisitos especiais a nível de conhecimento de código de CN e referências NCI.

7 – Conclusões e proposta de trabalhos futuros

As conclusões deste trabalho consistem na análise do que nele foi feito e na contraposição com os objetivos propostos.

As funcionalidades do CAM foram revistas e organizadas, e o seu enquadramento tecnológico foi abordado. Este objetivo foi atingido de uma forma geral, e em particular para a solução proposta para esta dissertação: Mastercam X5. A extensa utilização deste programa gerou critérios que permitem agora estender esta análise a outras soluções comerciais. Os exemplos expostos não mostraram funcionalidades que permitissem aferir, em última análise, o grau de automatização do *software* CAM. No entanto, deram uma resposta quanto à sua utilidade, e no que diz respeito ao planeamento tecnológico. O objetivo de avaliar as funcionalidades disponíveis neste tipo de aplicações foi, nesta perspetiva, parcialmente cumprido.

A avaliação da flexibilidade e possibilidades de configuração das aplicações CAM, e em particular para o Mastercam, foi cumprida. Não através de resultados de cariz científico, mas através da exemplificação de casos e através da sua configuração para uma máquina-ferramenta, inclusive com a construção de um pós-processador. Com esta última tarefa, ficaram avaliadas as possibilidades da configuração do CAM.

A configuração do Mastercam para uma máquina em particular – Okuma MC-40VA – foi feita na totalidade. A solução apresentada confere a capacidade esperada com este tipo de atividades. A menos de melhoramentos que possam ser incluídos neste "resultado", considera-se que foi um trabalho proveitoso, e que forneceu mensurável aprendizagem e ganho de competências.

Finalmente, e ainda acerca da averiguação do grau de automatização do Mastercam, relata-se que este objetivo não foi cumprido dentro dos planos propostos.

Ainda na linha desta conclusão, pretende-se dar resposta às questões colocadas no início desta dissertação:

- Qual e que tipo de contributo tem o utilizador neste processo?
- De que depende a qualidade dos programas gerados?

O suporte de uma resposta fundamentada e informada a estas perguntas, foi, como se constata neste capítulo, obtido ao longo das atividades desenvolvidas. Estas, assentes numa base de constante utilização e experimentação de uma oferta de CAM em particular, e mantendo todos os conceitos tecnológicos investigados presentes no momento, dividem-se em quatro grupos distintos, que definem o conhecimento ganho:

- Conhecimento do contexto onde está inserido o apoio ao planeamento tecnológico e apoio à tomada de decisão, e em que tipo de apoio consiste;
- Conhecimento do *software* Mastercam X5 na vertente de apoio ao planeamento tecnológico e à tomada de decisão;
- Conhecimento do potencial do CAM como apoio ao planeamento tecnológico e à tomada de decisão;
- Conhecimento dos requisitos da sua boa utilização, como fator influenciador do resultado final;

Qual o contributo que o utilizador tem na elaboração de código de CN?

O primeiro contributo que o utilizador tem no processo (obtenção de código de CN), é o da escolha do *software* usado. À luz do primeiro grupo "Conhecimento do contexto", também o utilizador deve saber que tipo de apoio precisa. Ou seja, nesta fase, assumindo que o único contributo a considerar é o que provoca um benefício, o conhecimento dos seus próprios requisitos, e do tipo de apoio oferecido no mercado CAM, é inicialmente o melhor contributo. Esta lógica explicita um pré-contributo. O segundo contributo passa agora para um lado mais técnico da questão. Aqui o utilizador deve ter *know-how* de processos de maquinagem: saber identificar orientações dominantes, saber relacionar estratégias de maquinagem e ferramentas e ter conhecimento das terminologias associadas às estratégias. Aqui este contributo é aplicado antes da escolha da operação a realizar. No que diz respeito à simulação da qualidade, esta não produz um resultado fiável o suficiente para que o utilizador não tenha de intervir criticamente. Assim sendo, existe aqui intervenção por parte do mesmo. Como se pode constatar nos exemplos das subsecções 4.1, 4.2 e 4.3, existe no Mastercam X5 constante interação com o utilizador. Posto isto, a intervenção técnica, no planeamento tecnológico, dá-se na escolha da operação, seleção da geometria e discernimento dos resultados, pelo que se conclui que aqui o contributo é técnico e sistemático. Interpretando agora a questão do ponto de vista do potencial do CAM, o contributo do utilizador pode ser variável. Exemplifique-se: na geração de código de CN, usando dois pós-processadores distintos com o mesmo *software*, sugere-se dois contributos distintos – um, maior, devido à edição, outro, quase nulo; no primeiro caso há uma fraca exploração do potencial do CAM, e no segundo caso há uma utilização desse potencial. O contributo aqui é pontual (deu-se na edição do pós-processador), e reflete a atitude exploratória que o utilizador deve ter. Por fim, à luz do conhecimento dos requisitos da melhor utilização do CAM, o contributo aqui assenta no conhecimento dos recursos envolvidos no processo, sejam materiais ou equipamentos. Logo, a utilização de bases-de-dados de ferramentas e materiais, assim como máquinas virtuais bem definidas, constitui um contributo técnico e inicialmente intenso, mas que terá uma manutenção mínima.

Resumindo esta análise, é possível dar uma resposta sintetizada à questão colocada:

O *know-how* de processos de fabrico com recurso maquinas-ferramenta é atualmente insubstituível, e contributos dessa natureza estarão presentes na utilização de qualquer *software* CAM atual. Esses contributos são técnicos, sistemáticos e formam um requisito da utilização de CAM. A diminuição do contributo exigido em função da configuração do sistema é um contributo pontual, fazendo com que o primeiro possa ser reduzido ou mesmo eliminado.

De que depende a qualidade dos programas gerados?

A qualidade dos programas gerados divide-se em dois focos de análise.

O primeiro é derivado de aspetos comuns do código de CN, e nele inserem-se as posições que o código descreve. A qualidade, sobre esta perspetiva, traduz-se no estado final da peça. Esta qualidade está relacionada com dois fatores essenciais: um diz respeito aos algoritmos desenvolvidos pela marca, e implementados no *software*. Estes algoritmos englobam tanto a geração de percursos, como a forma em que as curvas dos mesmos são discretizadas em segmentos de reta. O outro diz respeito à parametrização das operações. A previsibilidade da qualidade dos programas gerados, sob este foco de análise, depende da escolha do *software*, i.e., a escolha de um bom *software* pelo menos nesta área, levará a, previsivelmente, códigos com qualidade. Depende também do *know-how* e da transmissão dessa virtude através do domínio do *software*. Mais precisamente, da capacidade de reconhecer erros e saber como os corrigir e saber interpretar tolerâncias na discretização de curvas.

O segundo foco de análise corresponde aos aspetos particulares do código de CN. Aqui fala-se das funções modais e auxiliares específicas de um determinado controlador. Esta medida de qualidade traduz-se na utilização segura e previsível do equipamento. Um dos fatores que influenciam a qualidade, sob este foco, está relacionado com a informação disponibilizada para a completa configuração do programa: por exemplo se a informação é fidedigna e se não foram alterados nenhuns aspetos da máquina e/ou controlador de forma não documentada. Outro fator está diretamente ligado à capacidade de editar um pós-processador, que, como evidenciado, tem a possibilidade de garantir a total qualidade do programa gerado – mais uma vez, sob este foco em particular.

A melhor forma de responder à questão deste subsecção é relativizando o conceito de qualidade: qualidade é dada relacionando o que se pretende obter, e o que se obteve. Através deste raciocínio estabelece-se que um percurso gerado, previsivelmente bom, só é verdadeiramente validado quando é usado para produzir uma peça, e como tal, também a capacidade de correlacionar o percurso gerado com esse resultado final, é um fator do qual depende a qualidade dos percursos gerados. Logo, é possível concluir que, também a contínua utilização de um *software* em particular é um fator do qual depende a qualidade dos percursos gerados, pois só assim se adquire a experiência necessária para os correlacionar com a qualidade da peça final.

Agregando o raciocínio explanado, dá-se agora uma resposta sintetizada:

O experiente conhecimento da forma como os parâmetros das operações de maquinagem afetam a qualidade da peça final, assim como a qualidade do gerador de percursos e o grau de fiabilidade do pós-processador, associados ao *know-how* que permite validar o apoio à tomada de decisão, são os fatores dos quais depende a qualidade do código de CN gerado.

Trabalhos futuros

O trabalho desenvolvido nesta dissertação consistiu essencialmente, na caracterização de um estado tecnológico e na caracterização do seu uso. Como tal, ficam em aberto inúmeros aspetos da utilização propriamente dita do Mastercam X5 na maquinagem de uma peça real.

Mais ainda se acrescenta que neste trabalho não foram realizados testes com a máquina Okuma MC-40VA, pelo que seria interessante incluir esses ensaios como uma dessas propostas.

Como tal para proposta de trabalhos futuros, sugere-se:

- Teste do pós-processador com a Okuma MC-40VA;
- Projeto de sistema de fixação para peças com posicionamento de precisão para a máquina Okuma MC-40VA;
- Adaptação do Mastercam X5 a uma máquina de 5 eixos;
- Utilização de outros *software* e estudo comparativo;
- Otimização de estratégias com produção real e controlo metrológico.

Bibliografia

[1] C.-H. Chang and M. A. Melkanoff, NC Machine Programming and Software Design, Los Angeles, EUA: Prentice-Hall, 1989.

[2] H.-C. Zhang e L. Alting, Computerized Manufacturing Process Planing Systems, Londres, Reino Unido: Chapman & Hall, 1994.

[3] K. Wilson, "Manufacturing software can have a huge impact on your business, if you do it right - Learn how," CAD/CAM Connect Blog, 9 Janeiro 2013 . [Online]. Available: http://info.cadcamconnect.com/blog/bid/256571/Manufacturing-software-can-have-a-huge-impact-on-your-business-if-you-do-it-right-Learn-how. [Acedido em Março 2013].

[4] M. P. Groover e E. W. Zimmers Jr., CAD/CAM Computer-Aided Design And Manufacturing, Departement of Industrial Engineering Lehigh University, New Jersey: Prentice-Hall, Inc., 1984.

[5] Delcam, "CAD CAM Software Solutions - Delcam," 2013. [Online]. Available: http://www.delcam.com/index.asp. [Acedido em 2013].

[6] Mastercam, "Mastercam : Products," 2013. [Online]. Available: http://www.mastercam.com/Products/. [Acedido em 2013].

[7] Siemens, "NX for Manufacturing : Siemens PLM Software," 2013. [Online]. Available: http://www.plm.automation.siemens.com/en_us/products/nx/for-manufacturing/index.shtml. [Acedido em 2013].

[8] A. Christman, "Evolution of CAM Software," 2004. [Online]. Available: http://www.mmsonline.com/columns/evolution-of-cam-software. [Acedido em 2013].

[9] CAD CAM Fundamentals, "History of CAD/CAM," 2013. [Online]. Available: http://cadcamfunda.com/history_of_cadcam. [Acedido em 2013].

[10] P. Waurzyniak, "CAD/CAM Software Drives Innovation," Fevereiro 2010. [Online]. Available: http://www.sme.org/MEMagazine/Article.aspx?id=37628. [Acedido em 2013].

[11] P. Waurzyniak, "Betting Big on Multitasking," Fevereiro 2013. [Online]. Available: http://www.sme.org/MEMagazine/Article.aspx?id=71137. [Acedido em 2013].

[12] P. Waurzyniak, "CAD/CAM Software Advances," Fevereiro 2012. [Online]. Available: http://www.sme.org/MEMagazine/Article.aspx?id=49566. [Acedido em 2013].

[13] S. S. V. R. P. Radhakrishnan, CAD/CAM/CIM, New Age Internation, 2008.

[14] M. Albert, "Feature Recognition--The Missing Link To Automated CAM," 2001. [Online]. Available: http://www.mmsonline.com/articles/feature-recognition--the-

missing-link-to-automated-cam. [Acedido em 2013].

[15] HSMworks, "Integrated CAM for SolidWorks," 2013. [Online]. Available: http://www.hsmworks.com/overview/. [Acedido em 2013].

[16] D. Santos, "Is the CAM industry awakening?," Fevereiro 2013. [Online]. Available: http://camzone.org/2013/02/02/is-the-cam-industry-awakening/. [Acedido em 2013].

[17] K. C. Arabe, "CAD/CAM: Past, Present and Future," 2001. [Online]. Available: http://news.thomasnet.com/IMT/2001/02/23/cadcam_past_pre/. [Acedido em 2013].

[18] University of Arts, London, "University of Arts, London," [Online]. Available: http://www.arts.ac.uk. [Acedido em 2013].

[19] M. Tomljanovich, "CAD/CAM in the electronic manufacturing industry," Elsevier B.V., Genova, Itália, 1987.

[20] K. S. S. K. W. C. Wong KC, "Use of a patient-specific CAD/CAM surgical jig in extremity bone tumor resection and custom prosthetic reconstruction.," PubMed, Department of Orthopaedics and Traumatology, Prince of Wales Hospital, The Chinese University of Hong Kong, Shatin, Hong Kong, 2012.

[21] Manufacturing Engineering Media, "Shop Solutions: CAM Software Speeds Formula One Racers," 2006. [Online]. Available: http://www.sme.org/MEMagazine/Article.aspx?id=27458. [Acedido em 2013].

[22] A. Arbor, "CAM Software Market Leaders Named by CIMdata NC Market Analysis Report for 2010 Released," 2011. [Online]. Available: http://www.cimdata.com/news_events/press_release.html?press_release_ID=99. [Acedido em 2013].

[23] A. Arbor, "CIMdata Reports That Global NC Market Grew 14.4% in 2011," 2012. [Online]. Available: http://www.cimdata.com/news_events/press_release.html?press_release_ID=131. [Acedido em 2013].

[24] O. MC-40VA, Manual 3547-E, OPERATION & MAINTENANCE MANUAL, 1991.

ANEXO A : PROGRAMAS DE CÓDIGO DE CN GERADOS EM 6.2

CÓDIGO GERADO COM PÓS-PROCESSADOR ORIGINAL

```
%
O0000(MODEL1OLD)
(DATE=DD-MM-YY - 04-07-13
TIME=HH:MM - 20:55)
(MCX FILE - C:\...\*.MCX-5)
(NC FILE - C:\...\*.NC)
(MATERIAL-ALUMINUM MM-2024)
(T3|FRESA TOPO D40 Z4 HSS|H3)
(T15|FRESA CDG D25 Z3
HSS|H15)
N100 G21
N102 G0 G17 G40 G49 G80 G90
N104 T3 M6
N106 G0 G90 G55 X-70.189
Y13.644 S954 M3
N108 G43 H3 Z80.
N110 M98 P0001
N112 G90 X129.811 Y13.644
N114 M98 P0001
N116 G90 X-70.189 Y213.644
N118 M98 P0001
N120 G90 X129.811 Y213.644
N122 M98 P0001
N124 G90 X21.716 Y-103.137
N126 M98 P0002
N128 G90 X221.716
N130 M98 P0002
N132 G90 X21.716 Y96.863
N134 M98 P0002
N136 G90 X221.716
N138 M98 P0002
N140 M5
N142 G91 G28 Z0.
N144 M01
N146 T15 M6
N148 G0 G90 G55 X150.104
Y25.251 S1527 M3
N150 G43 H15 Z80.
N152 M98 P0003
N154 G90 X350.104 Y25.251
N156 M98 P0003
N158 G90 X150.104 Y225.251
N160 M98 P0003
N162 G90 X350.104 Y225.251
N164 M98 P0003
N166 M5
N168 G91 G28 Z0.
N170 G28 X0. Y0.
N172 M30

O0001
N100 G91
N102 Z-15.
N104 G1 Z-70. F190.8
N106 X39.951 Y-1.961 F381.6
N108 G3 X41.914 Y37.991 R40.
N110 G1 X.163 Y3.324
N112 X.882 Y4.417
N114 X1.822 Y4.471
N116 X2.448 Y3.889
N118 X2.885 Y3.294
N120 X3.188 Y2.749
N122 X3.414 Y2.271
N124 X3.598 Y1.854
N126 X3.759 Y1.482
N128 X3.898 Y1.137
N130 X4.013 Y.807
N132 X4.093 Y.484
N134 X4.133 Y.162
N136 X4.127 Y-.158
N138 X4.076 Y-.475
N140 X3.985 Y-.794
N142 X3.862 Y-1.115
N144 X3.713 Y-1.446
N146 X3.546 Y-1.799
N148 X3.355 Y-2.188
N150 X3.126 Y-2.623
N152 X2.828 Y-3.105
N154 X2.421 Y-3.615
N156 X1.864 Y-4.094
N158 X1.13 Y-4.367
N160 X.319 Y-3.91
N162 X-.163 Y-3.324
N164 X-.882 Y-4.417
N166 X-1.822 Y-4.471
N168 X-2.448 Y-3.889
N170 X-2.885 Y-3.294
N172 X-3.188 Y-2.749
N174 X-3.414 Y-2.271
N176 X-3.598 Y-1.854
N178 X-3.759 Y-1.482
N180 X-3.898 Y-1.137
N182 X-4.013 Y-.807
N184 X-4.093 Y-.484
N186 X-4.133 Y-.162
N188 X-4.127 Y.158
N190 X-4.076 Y.475
N192 X-3.985 Y.794
N194 X-3.862 Y1.115
N196 X-3.713 Y1.446
N198 X-3.546 Y1.799
N200 X-3.355 Y2.188
N202 X-3.126 Y2.623
N204 X-2.828 Y3.105
N206 X-2.421 Y3.615
N208 X-1.864 Y4.094
N210 X-1.13 Y4.367
N212 X-.319 Y3.91
```

```
N214 G3 X-43.128 Y36.606 R40.
N216 G1 X-39.867 Y-3.261
N218 G0 Z25.
N220 Z60.
N222 X-1.965 Y-73.019
N224 Z-15.
N226 G1 Z-75. F190.8
N228 X40. F381.6
N230 G3 X40. Y40. R40.
N232 G2 X86.568 R43.284
N234 X-86.568 R43.284
N236 G3 X-40. Y40. R40.
N238 G1 X-40.
N240 G0 Z25.
N242 Z65.
N244 X182.319 Y-185.
N246 Z-15.
N248 G1 Z-85. F190.8
N250 Y40. F381.6
N252 G3 X-40. Y40. R40.
N254 G1 X-84.035
N256 Y130.
N258 X84.334
N260 X7.642 Y-8.558
N262 X2.09 Y-2.824
N264 X1.883 Y-2.901
N266 X1.294 Y-2.058
N268 X.723 Y-1.086
N270 X.144 Y-.198
N272 X1.745 Y-.52
N274 X.611 Y-.175
N276 X2.219 Y-.678
N278 X2.086 Y-.738
N280 X1.954 Y-.805
N282 X2.146 Y-1.023
N284 X3.37 Y-1.983
N286 X3.29 Y-2.518
N288 X2.918 Y-2.847
N290 X2.532 Y-3.115
N292 X2.155 Y-3.326
N294 X1.806 Y-3.501
N296 X1.49 Y-3.662
N298 X1.207 Y-3.826
N300 X.95 Y-3.999
N302 X.707 Y-4.176
N304 X.468 Y-4.336
N306 X.216 Y-4.255
N308 X.023 Y-2.073
N310 X-.127 Y-4.33
N312 X-.307 Y-3.72
N314 X-.604 Y-4.419
N316 X-.82 Y-3.996
N318 X-.377 Y-1.502
N320 X-1.135 Y-3.772
N322 X-1.436 Y-3.644
N324 X-1.177 Y-2.456
N326 X-1.925 Y-3.332
N328 X-2.162 Y-2.968
N330 X-1.349 Y-1.589
N332 X-2.584 Y-2.611
N334 X-2.751 Y-2.191
N336 X-1.158 Y-.807
N338 X-2.841 Y-1.723
N340 X-2.944 Y-1.415
N342 X-3.253 Y-1.224
N344 X-1.532 Y-.487
N346 X-1.63 Y-.478
N348 X-1.193 Y-.343
N350 X-.56 Y-.175
N352 X-.436 Y-.595
N354 X-.163 Y-.244
N356 X-.745 Y-1.164
N358 X-.956 Y-1.528
N360 X-2.359 Y-3.572
N362 X-2.376 Y-3.007
N364 X-7.068 Y-7.527
N366 G3 X1.779 Y-56.541 R40.
N368 G1 X29.159 Y-27.381
N370 G0 Z100.
N372 M99

O0002
N100 G91
N102 Z-15.
N104 G1 Z-80. F190.8
N106 X28.284 Y28.284 F381.6
N108 G3 Y56.569 R40.
N110 G1 X-68.284 Y68.284
N112 X68.284 Y68.284
N114 X68.284 Y-68.284
N116 X-68.284 Y-68.284
N118 G3 Y-56.569 R40.
N120 G1 X28.284 Y-28.284
N122 G0 Z95.
N124 M99

O0003
N100 G91
N102 Z-15.
N104 G1 Z-80. F229.1
N106 X-17.678 Y17.678 F458.1
N108 G3 X-35.355 R25.
N110 G1 X-40. Y-40.
N112 G3 Y-35.355 R25.
N114 G1 X17.678 Y-17.678
N116 G0 Z25.
N118 Z70.
N120 X-124.853 Y124.853
N122 Z-15.
N124 G1 Z-80. F229.1
N126 X17.678 Y-17.678 F458.1
N128 G3 X35.355 R25.
N130 G1 X40. Y40.
N132 G3 Y35.355 R25.
N134 G1 X-17.678 Y17.678
N136 G0 Z95.
N138 M99
%
```

ANEXO B : PROGRAMAS DE CÓDIGO DE CN GERADOS EM 6.2

CÓDIGO GERADO COM PÓS-PROCESSADOR EDITADO

```
$MODELO1NEW.MIN%
O0000(MODELO1NEW)
(DATE=DD-MM-YY - 04-07-13 TIME=HH:MM
- 20:56)
(MCX FILE - C:\...\*.MCX-5)
(NC FILE - C:\...\*.NC)
(MATERIAL - ALUMINUM MM - 2024)
(LIMITES - CODIGOS DE ERRO:)
(XFDL - X FORA DE LIMITES)
(YFDL - Y FORA DE LIMITES)
(ZFDL - Z FORA DE LIMITES)
(LIMITES DOS EIXOS :)
(X [-381;381])
(Y [-205;205])
(Z [200;650])
(T3|FRESA TOPO D40 Z4 HSS|H3)
(T15|FRESA CDG D25 Z3 HSS|H15)
N100G21
N110G0G17G40G53G80G90
N120T3M6
N130G15H01
N140G0G90X-70.189Y13.644S954M3
N150G56H3Z80.
N160CALL OSUB1
N170G90X129.811Y13.644
N180CALL OSUB1
N190G90X-70.189Y213.644(<YFDL)
N200CALL OSUB1
N210G90X129.811Y213.644(<YFDL)
N220CALL OSUB1
N230G90X21.716Y-103.137
N240CALL OSUB2
N250G90X221.716
N260CALL OSUB2
N270G90X21.716Y96.863
N280CALL OSUB2
N290G90X221.716
N300CALL OSUB2
N310M5
N320G91G30 P0
N330M01
N340T15M6
N350G0G15H01
N360X150.104Y25.251S1527M3
N370G56H15Z80.(<ZFDL)
N380G90
N390CALL OSUB3
N400G90X350.104Y25.251
N410CALL OSUB3
N420G90X150.104Y225.251(<YFDL)
N430CALL OSUB3
N440G90X350.104Y225.251(<YFDL)
N450CALL OSUB3
N460M5
N470G91G30 P0
N480M30

OSUB1
N100G91
N110Z65.Z-15.
N120G1Z-5.Z-70.F190.8
N130X39.951Y-1.961F381.6
N140G3X1.962Y-.048R40.
N150X39.952Y38.039R40.
N160G1X.163Y3.324
N170X.882Y4.417
N180X1.822Y4.471
N190X2.448Y3.889
N200X2.885Y3.294
N210X3.188Y2.749
N220X3.414Y2.271
N230X3.598Y1.854
N240X3.759Y1.482
N250X3.898Y1.137
N260X4.013Y.807
N270X4.093Y.484
N280X4.133Y.162
N290X4.127Y-.158
N300X4.076Y-.475
N310X3.985Y-.794
N320X3.862Y-1.115
N330X3.713Y-1.446
N340X3.546Y-1.799
N350X3.355Y-2.188
N360X3.126Y-2.623
N370X2.828Y-3.105
N380X2.421Y-3.615
N390X1.864Y-4.094
N400X1.13Y-4.367
N410X.319Y-3.91
N420X-.163Y-3.324
N430X-.882Y-4.417
N440X-1.822Y-4.471
N450X-2.448Y-3.889
N460X-2.885Y-3.294
N470X-3.188Y-2.749
N480X-3.414Y-2.271
N490X-3.598Y-1.854
N500X-3.759Y-1.482
N510X-3.898Y-1.137
N520X-4.013Y-.807
N530X-4.093Y-.484
N540X-4.133Y-.162
N550X-4.127Y.158
N560X-4.076Y.475
N570X-3.985Y.794
N580X-3.862Y1.115
N590X-3.713Y1.446
N600X-3.546Y1.799
N610X-3.355Y2.188
N620X-3.126Y2.623
N630X-2.828Y3.105
N640X-2.421Y3.615
N650X-1.864Y4.094
N660X-1.13Y4.367
N670X-.319Y3.91
N680G3X-39.867Y36.739R40.
N690X-3.261Y-.133R40.
N700G1X-39.867Y-3.261
N710G0Z20.Z25.
N720Z60.
N730X-1.965Y-73.019
N740Z65.Z-15.
N750G1Z-10.Z-75.F190.8
N760X40.F381.6
N770G3X40.Y40.R40.
N780G2X43.284Y43.284R43.284
N790X43.284Y-43.284R43.284
N800X-43.284Y-43.284R43.284
N810X-43.284Y43.284R43.284
N820G3X-40.Y40.R40.
N830G1X-40.
```

```
N840G0Z15.Z25.
N850Z65.
N860X182.319Y-185.
N870Z65.Z-15.
N880G1Z-20.Z-85.F190.8
N890Y40.F381.6
N900G3X-40.Y40.R40.
N910G1X-84.035
N920Y130.
N930X84.334
N940X7.642Y-8.558
N950X2.09Y-2.824
N960X1.883Y-2.901
N970X1.294Y-2.058
N980X.723Y-1.086
N990X.144Y-.198
N1000X1.745Y-.52
N1010X.611Y-.175
N1020X2.219Y-.678
N1030X2.086Y-.738
N1040X1.954Y-.805
N1050X2.146Y-1.023
N1060X3.37Y-1.983
N1070X3.29Y-2.518
N1080X2.918Y-2.847
N1090X2.532Y-3.115
N1100X2.155Y-3.326
N1110X1.806Y-3.501
N1120X1.49Y-3.662
N1130X1.207Y-3.826
N1140X.95Y-3.999
N1150X.707Y-4.176
N1160X.468Y-4.336
N1170X.216Y-4.255
N1180X.023Y-2.073
N1190X-.127Y-4.33
N1200X-.307Y-3.72
N1210X-.604Y-4.419
N1220X-.82Y-3.996
N1230X-.377Y-1.502
N1240X-1.135Y-3.772
N1250X-1.436Y-3.644
N1260X-1.177Y-2.456
N1270X-1.925Y-3.332
N1280X-2.162Y-2.968
N1290X-1.349Y-1.589
N1300X-2.584Y-2.611
N1310X-2.751Y-2.191
N1320X-1.158Y-.807
N1330X-2.841Y-1.723
N1340X-2.944Y-1.415
N1350X-3.253Y-1.224
N1360X-1.532Y-.487
N1370X-1.63Y-.478
N1380X-1.193Y-.343
N1390X-.56Y-.175
N1400X-.436Y-.595
N1410X-.163Y-.244
N1420X-.745Y-1.164
N1430X-.956Y-1.528
N1440X-2.359Y-3.572
N1450X-2.376Y-3.007
N1460X-7.068Y-7.527
N1470G3X-10.84Y-27.381R40.
N1480X12.619Y-29.16R40.
N1490G1X29.159Y-27.381
N1500G0Z100.
N1510RTS

OSUB2
N100G91
N110Z65.Z-15.
N120G1Z-15.Z-80.F190.8
N130X28.284Y28.284F381.6
N140G3X11.716Y28.284R40.
N150X-11.716Y28.285R40.
N160G1X-68.284Y68.284
N170X68.284Y68.284
N180X68.284Y-68.284
N190X-68.284Y-68.284
N200G3X-11.716Y-28.285R40.
N210X11.716Y-28.284R40.
N220G1X28.284Y-28.284
N230G0Z95.
N240RTS

OSUB3
N100G91
N110Z65.Z-15.
N120G1Z-15.Z-80.F229.1
N130X-17.678Y17.678F458.1
N140G3X-17.677Y7.322R25.
N150X-17.678Y-7.322R25.
N160G1X-40.Y-40.
N170G3X-7.322Y-17.678R25.
N180X7.322Y-17.677R25.
N190G1X17.678Y-17.678
N200G0Z10.Z25.
N210Z80.Z70.
N220X-124.853Y124.853
N230Z65.Z-15.
N240G1Z-15.Z-80.F229.1
N250X17.678Y-17.678F458.1
N260G3X17.677Y-7.322R25.
N270X17.678Y7.322R25.
N280G1X40.Y40.
N290G3X7.322Y17.678R25.
N300X-7.322Y17.677R25.
N310G1X-17.678Y17.678
N320G0Z80.Z95.
N330RTS
%
```

ANEXO C : EXCERTOS DO PÓS-PROCESSADOR MODIFICADO

Inicialização das variáveis e *strings* implementadas

Nesta primeira parte apresentar-se-ão as variáveis implementadas. Onde se entende necessário dar-se-á uma curta explicação, devidamente destacada.

As inicializações de variáveis é feita na forma **VARIABLE** : X, em que X é o seu valor de inicialização. As *strings* são inicializadas na forma **STRING** : "X", em que aqui X representa a *string* de inicialização escolhida. Todas as *strings* devem começar com a letra S.

- Valores customizáveis – usados no ficheiro de controlo.

```
# Valores customizáveis:
# ------------------------------------------------------------
# Inteiros:
```
(Estas variáveis só podem assumir valores inteiros)
```
# mi1 - Modo de sistema de sistemas de coordenadas
```
(Esta variável do ficheiro de controlo permite escolher o tipo de definição de sistemas de eixos. Embora já existisse, foi adaptada para usar **G15** em vez de **G54-59** (como será possível ver adiante))
```
#         1 = Usar G15: H01-H50 | H101-H150 -> com rotina de erro
```
(De acordo com o controlador OSP5020M, os valores de **G15** só podem ser usados na gama representada: de **H01** a **H50**, e de **H101** a **H150**. Valores definidos fora desta gama resultarão em mensagem de erro.)
```
#         2 = Usar G92
```
(Esta opção já estava definida anteriormente. No entanto, tentou-se implementar a utilização da função **VAPA***, do controlador OSP5020M. Esta função permite registar a posição momentânea dos eixos X, Y Z, podendo ser usado para iniciar um programa de CN de forma mais expedita. Por falta de testes, esta função não foi totalmente implementada.)
```
# mi2 - Programação em modo absoluto ou incremental
```
(Esta função já se encontra definida de origem no pós-processador.)
```
#         0 = absoluto
#         1 = incremental
```

```
# mi3 - Ponto casa (de retorno) definido por G28 ou G30
```
(Esta função já se encontra definida de origem no pós-processador, necessitando só de um acrescento à função modal **G30**, que permite definir 5 posições casa distintas. No entanto, usar-se-á apenas a **P0**, que é imutável e que diz respeito à posição neutra dos eixos (0,0,650).)
```
#         0 = G289
#         1 = G30
# mi4 - Erros de saída de limites interrompem post (provoca a paragem do pós processamento em caso de
erro)
#         0 = Não  (Esta escolha não permite interromper o pós-processamento em caso de erro, deixando-o funcionar até ao fim.)
#         1 = Sim  (Esta escolha permite interromper o pós-processamento em caso de erro, poupando tempo em códigos longas.)
# mi5 - Reservado  (Esta posição ficou reservada para futuras implementações.)
# mi6 - Função VAPA* - Value Of Actual Position of Axis X Y Z (A DESENVOLVER, inibido no POSPROC.)
#        0 = Não
#        1 = Sim
```
(A função **VAPA*** - *Value of Actual Position of Axis* – permite, no momento da sua chamada, emitir a posição de um eixo, X, Y ou Z. Usando em conjugação com o código modal **G92**, e colocando a ferramenta na posição pretendida de início de operação, esta função poderá acelerar o arranque dos programas)

- Inicialização de variáveis e *strings*.

```
# ------------------------------------------------------------
#variáveis e strings específicos deste pós-processador
# ------------------------------------------------------------
```
(Em baixo são inicializadas as variáveis dos valores customizáveis implementados a partir de **mi4$** (**mi1$**, **mi2$** e **mi3$** são inicializadas noutro local).)
```
mi4$ : 0          # opção de terminar pós-processador durante verificação de limites
mi5$ : 0          # RESERVADO
mi6$ : 0          # VAPA* (EXPERIMENTAL)
```
(Estas variáveis abaixo foram desenvolvidas para utilização com o VAPA, e de momento, não estão a ser utilizadas.)
```
x_init        : 0 # posição inicial x (opcional)
y_init        : 0 # posição inicial y (opcional)
z_init        : 0 # posição inicial z (opcional)
```
(As variáveis seguintes dizem respeito às rotinas de verificação de erros de programação fora do limite dos eixos X, Y, Z.)
(As variáveis do tipo "**limits**" iniciam-se com o valor 0, e são depois alteradas por uma rotina de leitura das definições da máquina virtual. Estas variáveis conterão nessa altura os valores dos limites dos eixos.)

(As variáveis "erro" são *flag*s de erro que ficam ativas quando é detetado um erro.)

```
x_limits_min : 0 # limite x inferior
x_limits_max : 0 # limite x superior
errox : 0        # flag de erro
y_limits_min : 0 # limite y inferior
y_limits_max : 0 # limite y superior
erroy : 0        # flag de erro
z_limits_min : 0 # limite z inferior
z_limits_max : 0 # limite z superior
erroz : 0        # flag de erro
```

(As variáveis seguintes registam a posição X, Y e Z anterior a uma dada que está a ser escrita num ficheiro de código CN (para efeitos de comparação).)

(A sua função será melhor explicada durante a revisão da implementação do detetor de erros de curso.)

```
xabs_b : 0     # valor absoluto anterior de X
yabs_b : 0     # valor absoluto anterior de Y
zabs_b : 0     # valor absoluto anterior de Z
```

(As variáveis representadas abaixo dizem respeito à obtenção da ID dos componentes na árvore da máquina virtual, para extração de parâmetros.)

(O seu funcionamento será melhor explicado durante a revisão da implementação do extrator de parâmetros da máquina.)

(Algumas destas variáveis são apenas usadas para efeitos de *debugging*).

```
comp_n2: 0     # ID do componente (para efeitos de debugging)
comp_n : 0     # ID do componente
comp_i : 0     # ID do componente inicial no NCI
comp_f : 0     # ID do componente final no NCI
ent_i  : 0     # entidade inicial para scan de variáveis
```

(Estas *strings* que se seguem são usadas para comparar nomes de IDs dos componentes da árvore da máquina virtual.)

(O seu funcionamento será melhor explicado durante a revisão da implementação do extrator de parâmetros da máquina.)

```
saxis_lbl : "" # label do eixo
strx    : "X"  # string que representa X
stry    : "Y"  # string que representa Y
strz    : "Z"  # string que representa Z
```

(As variáveis seguintes são *flags* usadas no registo de parâmetros associados aos eixos.)

```
codestart : 0  #flag usada para permitir a entrada na função de procura de parâmetros
x_set : 0      #flag usada na procura de dados do eixo X
```

```
y_set : 0       #flag usada na procura de dados do eixo Y
z_set : 0       #flag usada na procura de dados do eixo Z
```
(Esta variável que se representa abaixo diz respeito à numeração de sub-rotinas.)
```
osub_prog_n  : 0 #designação do programa da subrotina
```
(As variáveis abaixo dizem respeito à extração de informações relativas às ferramentas.)
```
tool_ol       :0 #tool overall length - comprimento total da ferramenta sem fixação
tool_hl       :0 #tool holder length - comprimento da fixação
tool_tl       :0 #tool total length = tool_ol+tool_hl
tool_d        :0 #tool diâmeter - diâmetro da ferramenta
fto           :0 #first tool offset - offset da 1ª ferramenta
```

(A *string* seguinte é usada no início do código de CN, para o formatar conforme as necessidades do controlador OSP5020M.)
```
sfilename     : "" #string do nome do ficheiro
```
(As *strings* abaixo representadas são usadas no chamamento e retorno de sub-rotinas.)
```
strosub       : "OSUB" #chamamento de subrotina
strrts        : "RTS"  #retorno de subrotina
# ------------------------------------------------------------
#conjunto de mensagens de erro de programação fora de limites
# ------------------------------------------------------------
```
(Esta *strings* abaixo representadas são usadas para marcar erros de curso no código de CN, assim como definir os códigos de erro no cabeçalho.)
```
serrox        : ""
serrox_y      : "(XFDL - X FORA DE LIMITES)"
serroy        : ""
serroy_y      : "(YFDL - Y FORA DE LIMITES)"
serroz        : ""
serroz_y      : "(ZFDL - Z FORA DE LIMITES)"
serroexit     : "(VERIFICAR LIMITES DO PROGRAMA:POST TERMINADO)"

# ------------------------------------------------------------
```
#Seleção do mês (As *strings* aqui presentes cumprem só a função de mostrar o mês no cabeçalho. Foram apenas traduzidas do inglês para português.)
```
smon0    : ""
```

```
smon1   : "JAN."
smon2   : "FEV."
smon3   : "MAR."
smon4   : "ABR."
smon5   : "MAI."
smon6   : "JUN."
smon7   : "JUL."
smon8   : "AGO."
smon9   : "SET."
smon10  : "OUT."
smon11  : "NOV."
smon12  : "DEC."
smonth  : ""          #Target string
```

As linhas seguintes representam a definição de formatos para as variáveis inicializadas antes. Os números nelas presentes representam um estilo de formatação. Tratando-se de algo pouco relevante, não se irá dar mais atenção a este bloco.

```
#formatos para as variáveis implementadas
fmt 4 x_limits_min
fmt 4 x_limits_max
fmt 4 y_limits_min
fmt 4 y_limits_max
(…)
fmt 2 tool_tl
fmt 11 x_init
fmt 11 y_init
fmt 11 z_init
```

Apresentadas as variáveis e *strings* que se incluíram no pós-processador, em virtude das alterações ao mesmo, constata-se que ambas se dividem em:

- Variáveis e *strings* usadas nas rotinas de deteção de erros de curso;
- Variáveis e *strings* usadas nas rotinas de extração de parâmetros da máquina virtual;
- Variáveis e *strings* usadas em funções auxiliares, como o chamamento de sub-rotinas, ou especificação da extensão do ficheiro do código de CN.

Esta distribuição permite verificar à partida o tipo de alterações que foram implementadas no pós-processador.

Outras *strings*, que foram alteradas, e que se omitiram da lista acima apresentada, dizem respeito a funções modais. Aqui algumas foram implementadas de origem, como é o caso da "sg56", "sg15" ou "sg43". Outras foram simplesmente alteradas, como é o caso da "sg30", e da "sg99":

```
sg56      : "G56"      #Compensação do comprimento da ferramenta (sentido Z, usado para offset de
trabalho)
sg15      : "G15"      #Sistema de eixos do tipo H1-50, H101-150
sg43      : "G43"      #Cancelamento compensação tridimensional
sg30      : "G30 P0"   #Retorno à posição CASA, ponto P0 = posição neutra dos eixos
sg99      : "G71"      #Posição de retorno de ciclos de furação (No FANUC era G99)
```

Na fase seguinte, alteração aos blocos de começo de ficheiro, ir-se-á rever as alterações à escrita do cabeçalho.

Alteração aos blocos de começo de ficheiro

Os blocos de começo de ficheiro são responsáveis pelo cabeçalho do ficheiro de código de CN. Tipicamente nestes encontram-se funções de escrita de informações como a data e hora, nomes de ficheiros e materiais usados.

As alterações realizadas a este código consistem implementação de uma rotina de escrita da extensão do ficheiro, de acordo com o OSP5020M, escrita dos limites do curso dos eixos, assim como explicação das mensagens de erro que poderão figurar no código de CN gerado.

Analisando este excerto:

```
pheader$   #Call before start of file
```
(**pheader$** é a função chamada pelo **MP.dll** para iniciar a escrita do cabeçalho.)
(As duas próximas linhas tratam da escrita do nome do ficheiro de código de CN, com a extensão usada pelo OSP5020M - ***.MIN** (como antes apontado).) (Aqui há apenas um processo de adição de *strings*. Funções notáveis são o **sprogname$**, que é uma função-*string* pré-definida representativa do nome do ficheiro e **e$**, que confirma a escrita da linha em questão no ficheiro de código de CN (i.e., **e$=output(sfilename),** em que **sfilename = "$NOMEFICHEIRO.MIN%**")

```
      sfilename = "$" + sprogname$ + ".MIN" + "%"
      sfilename, e$
      (…)
```

```
      sopen_prn, "DATE=DD-MM-YY - ", date$, " TIME=HH:MM - ", time$, sclose_prn, e$
```
Aqui é feita a escrita da data e hora.)

```
      (…)
      sopen_prn, "MCX FILE - ", *smcpath$, *smcname$, *smcext$, sclose_prn, e$
```
(Aqui escreve-se o local do ficheiro MCX-5.)

```
      sopen_prn, "NC FILE - ", *spathnc$, *snamenc$, *sextnc$, sclose_prn, e$
```
(Aqui escreve-se o local do ficheiro de CN.)

```
      sopen_prn, "MATERIAL - ", *stck_matl$, sclose_prn, e$
```
(Aqui escreve-se a informação acerca do material.)

```
      sopen_prn, "LIMITES - CODIGOS DE ERRO:", sclose_prn, e$
```
(Aqui introduz-se a escrita das mensagens de erro.)

```
      *serrox_y, e$
```
("<XFDL – X FORA DE LIMITES")

```
      *serroy_y, e$
```
("<YFDL – Y FORA DE LIMITES"),

```
      *serroz_y, e$
```
("<ZFDL – Z FORA DE LIMITES". O asterisco * indica que a variável ou *string* está a ser "forçada"[58] para escrita.)

```
      codestart = 1
```
(Aqui ativa-se a *flag* de início de escrita do programa de CN.)

```
      plimits
```
(Aqui é chamada a função implementada neste pós-processador para iniciar a extração dos limites dos eixos: **plimits**.)

(As linhas que se seguem dizem respeito à escrita dos limites)

(Neste conjunto, **sopen_prn** introduz a escrita de abertura parêntesis (**sclose_prn** introduz a escrita do oposto). **e$** ordena o output das linhas.)

(Entre a escrita de parêntesis encontram-se as variáveis associadas ao limite dos eixos.)

```
      sopen_prn, "LIMITES DOS EIXOS :", sclose_prn, e$
      sopen_prn, "X ["*x_limits_min, ";" *x_limits_max, "]"sclose_prn, e$
      sopen_prn, "Y ["*y_limits_min, ";" *y_limits_max, "]"sclose_prn, e$
      sopen_prn, "Z ["*z_limits_min, ";" *z_limits_max, "]"sclose_prn, e$
```
(A função **plimits** tem um funcionamento que é explicado ao longo deste ponto.)

```
plimits
      if ent_i>0,
      #*comp_i, e$ #linha usada para debugging
```
(Esta linha é só usada para *debugging*[59], e não é lida pela aplicação **MP.dll**.)

```
      [rd_mch_ent_no$ = ent_i
      rd_md$]
```

[58] Este aspeto do mecanismo de programação ainda é relativamente desconhecido.

[59] Este termo é usado para designar um processo em que se eliminam os erros de programação de um determinado programa.

Para melhor explicar o funcionamento da função **plimits**, comece-se pelo estudo deste excerto que segue, de um ficheiro NCI (i.e., CLFILE). Componentes definidos numa árvore de uma dada máquina virtual têm os seus parâmetros associados aos códigos da classe **20600**:

```
(…)
20600
87 1C4C89B8-FBBA-11DA-991B-001111AFFC59 "Y" "Y" Y Axis
20600
88 1C4C89BA-FBBA-11DA-991B-001111AFFC59 "X" "X" X Axis
(…)
```

A cada código **20600**, estão associados alguns parâmetros associados ao objeto em foco. Na segunda linha deste exemplo, o primeiro parâmetro – 87 – é a ID do componente do eixo Y na árvore da máquina virtual. Este parâmetro nunca se repete de uma forma previsível entre vários modelos *.mcx-5, mesmo que partilhem a mesma máquina virtual. No entanto, para extrair informações deste componente de forma automática, é necessário conhecer essa ID.

O intuito da função **plimits** é o de iniciar o processo de procura de IDs. Quando é chamada, verifica a condição "**ent_i** > 0"(**ent_i** – entidade inicial).

O valor da variável **comp_n** é atualizado a cada leitura do ficheiro NCI. A função seguidamente apresentada, **pparameter$** faz um varrimento inicial do ficheiro NCI. A função, para este pós-processador, foi modificada para procurar parâmetros associados ao código **20600** (evidenciado a azul). Porque a função **rpar** lê o 1º parâmetro associado a um dado código – neste caso o **20600** – quando essa procura encontra um parâmetro que corresponde ao código **20600**, a variável **comp_n** assume o valor não nulo da ID associada. Isto satisfaz a condição evidenciada a amarelo. O que depois sucede é que a variável **comp_f** (componente final) assume o valor da variável **comp_n** (componente n). De acordo com a 3ª linha, a condição também é satisfeita, pois **comp_i** foi inicializado com o valor nulo, e agora **comp_n** possui um valor maior que zero. O que esta rotina faz é a definição do ID do componente inicial (**comp_i**) e um ID final (**comp_f**) ao longo dos varrimentos da função **pparameter$**, definindo os IDs que estão atribuídos à árvore da máquina virtual. Paralelamente a condição "**ent_i** > 0", devido a "**ent_i** = **comp_i**", fica satisfeita, e a função **plimits** pode proceder.

```
pparameter$     #Read operation parameters
#rd_params is used to call pparameter postblock and read the parameters of the operation specified in
rd_param_op_no
#"pparameter", ~prmcode$, ~sparameter$, e$
```

```
if prmcode$ = 20600, comp_n = rpar(sparameter$, 1)  (Leitura de parâmetros da classe 20600 e atualização da variável comp_i .)
      if (comp_n>0), [comp_f=comp_n]  #, e$ # descomentar ", e$" para debugging
      if (comp_i=0 & comp_n>0), [comp_i = comp_n, ent_i = comp_i]  (Definição de comp_i e ent_i.)
```

Cumprida a condição "**comp_i** > 0", são chamadas duas funções pré-definidas no MP.dll:

- **rd_mch_ent_no$**
- **rd_md$**

Esta função – *read machine entity number* – como o nome indica, lê uma entidade com um código particular. Esse mesmo código foi já definido pela variável **ent_i**, e que agora corresponde ao ID do primeiro componente no ficheiro NCI. Isto é, a instrução "**rd_mch_ent_no$** = **ent_i"** significa que a função **rd_md$** – que significa *read machine definition* – irá ler os parâmetros do componente de ID = **ent_i** = **comp_i**. A função **rd_md$** chama uma função designada **pmachineinfo$**.

Esta função – **pmachineinfo$** – tem uma vocação facilmente adivinhável. Ela encontra-se num local particular do pós-processador, que não deve ser alterado. A continuação da função **plimits** passará então para lá, seguindo com a lógica que se apresenta comentada:

- O código **17390**, em baixo, é usado para aceder a um dos parâmetros de componentes da máquina virtual. Especificamente, para ler a informação guardada na *string* da etiqueta usada para definir qual o eixo que lhe está associado. Porque este pós-processador diz respeito a uma máquina específica, o componente que se procura é do tipo eixo linear. Como tal, o código **17390** só pode ter associado a si 1 de 3 *string*s: X, Y ou Z. Recorde-se que o componente que está a ser lido pela função **rd_mch_ent_no$** é definido pela "ID = **ent_i**".

```
pmachineinfo$

      if codestart = 1,  (Aqui é verificado se o programa já começou a ser escrito.)
      [if prmcode$ = 17390, saxis_lbl = sparameter$   a  A string saxis_lbl guardará aqui o valor da extração do parâmetro
17390.
       if x_set = 0,  Se x_set = 0, então o valor dos limites do eixo X ainda não foram gravados.
            [
            if (saxis_lbl = strx), Aqui essa string é comparada com a string strx que corresponde simplesmente a "X".
                  [
```

```
                if prmcode$ = 17726, [x_limits_max = rpar(sparameter$, 1), x_set = 1]
                if prmcode$ = 17725, x_limits_min = rpar(sparameter$, 1)
```
(Nas duas linhas acima são acedidos os códigos **17726** e **17725**, e gravados os valores nas variáveis correspondentes.)
(Nas mesmas linhas são também ativadas as *flags* dos eixos, e é feito o *reset* à variável **ent_i**.)
```
                ]
            ]
            (…)
```

A condição "**x_set** = 0?", assim como as comparações entre *string*s, são repetidos para os outros 2 eixos (Y e Z). Os códigos **17726** e **17725** estão associados às definições da máquina virtual, mais especificamente aos limites dos eixos dos componentes apontados pela ID gravada em **ent_i**. Caso nenhuma condição seja satisfeita ao longo desta lógica, o valor de **ent_i** é incrementado em uma unidade, e **plimits** é chamado outra vez. Caso a igualdade de *string*s se verifique para um dos eixos, e caso o parâmetro do código **17726** tenha um valor não nulo, o seu valor será gravado em ***_limits_max**, e o valor do parâmetro do código **17725** em ***_limits_min**. Feito isto, a *flag* ***_set** é igualada a 1, acusado o registo de limites em X, Y ou Z. Após isso, dá-se na mesma o incremento de **ent_i**, e a função é reciclada.

```
            (…)
            ]
    ]
    ]
    if (x_set=0|y_set=0|z_set=0),  (Aqui é verificado se ainda falta gravar valores de limites de eixos, através dos valores das flags.)
          [ent_i=ent_i+1, plimits]  (Por fim nesta linha incrementa-se, em uma unidade, o valor da variável ent_i, e reinicia-se o ciclo.)
    else,
```

Esta explicação conclui o segundo ponto da exposição do pós-processador. As funcionalidades instaladas nas rotinas iniciais do pós-processador conseguem, com sucesso, extrair informações acerca dos limites de curso dos eixos lineares e organizá-los de modo a estarem dispostos no início do ficheiro de código de CN. Preferiu-se esta organização pois desta forma, em caso de deteção de erros, as informações dos limites dos eixos estão visíveis.

Esta funcionalidade foi otimizada para a utilização da interface do Mastercam, sendo apenas necessário definir na árvore da máquina virtual, nos componentes de movimento linear, os limites de curso – sendo desnecessário a edição do pós-processador para fazer a alteração dos mesmos. Este sistema permite também qualquer organização da árvore de componentes, desde que exista pelo menos um eixo linear.

De referir ainda é a pequena alteração ao início do cabeçalho, que agora contempla a extensão do ficheiro de código de CN, de acordo com o controlador OSP5020M.

No próximo ponto verificar-se-ão as alterações feitas aos blocos de pós-processamento propriamente dito. Neste ponto enquadram-se as rotinas de escrita das posições dos eixos, e as rotinas implementadas para efeitos de deteção de erros.

Alteração aos blocos de pós-processamento geral

As alterações apresentadas neste ponto, como foi mencionado, resumem-se à implementação de rotinas que detetem erros relacionados com a programação fora dos limites da máquina. Outras alterações neste ponto são mais triviais, como a substituição de alguns códigos usados em controladores FANUC, pelos que se usam com o OSP5020M.

Apesar do ambiente de simulação permitir o controlo dos limites da máquina, este é baseado no percurso gerado pelo Mastercam, e não no código de CN em si. Como tal, a deteção de erros no código de CN funciona como uma última barreira de segurança, alertando para os erros do código possa apresentar. Mais ainda, no caso de se fazer uso da interrupção do pós-processamento ao se detetarem erros.

De seguida exemplifica-se o início dos blocos de pós-processamento geral, com a escrita das primeiras funções modais:

```
psof$      #Start of file for non-zero tool number –  (Início do programa de CN com ferramenta não nula (ou seja, que existe).)

      if ra_error = 1, [if mprint(sra_error, 2) = 2, exitpost$]  (Verificação de existência de erros (original).)
      toolchng = one  (Aqui é ativada a flag toolchng, para mudança de ferramenta no início do programa.)
      if ntools$ = one,
      [
      #skip single tool outputs, stagetool must be on
      stagetool = m_one
      !next_tool$
      ]
      pbld, n$, *smetric, e$  (Escolha do sistema métrico (smetric), e output (e$).)
      pbld, n$, *sgcode, *sgplane, scc0, "G53", sg80, *sgabsinc, e$
```

(Na linha em cima note-se a escolha do modo de avanço (**sgcode**), plano de trabalho (**sgplane**), cancelamento da compensação de diâmetro (**scc0**), cancelamento do *offset* da ferramenta (alteração efectuada, de **G49** para **G53**), cancelamento de modo de funcionamento de ciclos embutidos (**sg80**), e seleção do modo absoluto ou relativo (**sgabsinc**). As funções modais que se apresentam como **sgcode** (ao contrário de **G00**, como seria de esperar) e **sgplane**, dependem de escolhas feitas nos parâmetros do ficheiro de controlo, ou das operações.)

```
sav_absinc = absinc$
if mi1$ = two, #Work coordinate system
```
(Aqui verifica-se que modo de sistema de coordenadas foi escolhido no ficheiro de controlo.)
```
[
absinc$ = one
#pfbld, n$, sgabsinc, *sg28ref, "Z0.", e$
#pfbld, n$, *sg28ref, "X0.", "Y0.", e$
```
(Estas duas linhas cima (evidenciadas a amarelo), de origem, vieram inibidas. Estas linhas dizem respeito à função de retorno, no modo de funcionamento com coordenadas no referência "peça" **G92**. No entanto, manter-se-ão assim, pois a forma de funcionamento dos sistemas de eixos com o controlador OSP5020M necessitará de um bloco novo por si só.)
```
if mi6$ = 1,
[
#pbld, n$, sg92, "VAPAX", "VAPAY", zh$, e$
]
if mi6$ = 0,
[
pbld, n$, sg92, *xh$, *yh$, *zh$, e$
]
absinc$ = sav_absinc
]
```
(Nas linhas acima representadas está a tentativa de implementação do **VAPA*** (inibida, com a colocação do símbolo #, que invalida a linha do pós-processador). No entanto, nas últimas 6 linhas, permite-se a utilização do sistema de coordenadas peça com o código modal **G92**, que utiliza a posição X, Y e Z de referência (**xh$**, **yh$** e **zh$**). A linha em causa escreve essa função modal e a sua posição.)
```
pcom_moveb
```
(Esta linha chama as funções de leitura do NCI e preparação para escrita de posições.)
```
Pcheckaxis
```
(Esta linha verifica a existência de um eixo rotativo. Aqui, caso haja um na máquina virtual, será emitida uma mensagem de erro.)
```
c_mmlt$ #Multiple tool subprogram call
ptoolcomment
```
(Aqui expõe-se comentários acerca da ferramenta para a operação que será realizada a partir deste ponto (opcional).)
```
comment$
```

```
      pcan  (Esta linha, caso a operação o exija, procede ao chamamento de um ciclo embutido.)
      pbld, n$, *t$, sm06, e$  (Aqui escrevem-se os códigos apropriados para uma troca de ferramenta.)
      if mi1$ = one, absinc$ = zero
      pcan1, pbld, n$, pwcs, e$  (Nesta linha chama-se a função pwcs, modificada para os sistemas de coordenadas no formato G15 Hxx.)
      pbld, n$, *sgcode, *sgabsinc, pfxout, pfyout,  (Aqui a ferramenta é colocada na posição de início da operação.)
            [if nextdc$ <> 7, *speed, *spindle], pgear, strcantext, e$  (O output dá-se nesta linha)
(Nas linhas abaixo, escrevem-se os códigos modais para a compensação da ferramenta (alterado para G56), com o offset de acordo com o controlador
OSP5020M, o posicionamento da ferramenta em Z (que só é feito após a definição do offset), e os códigos de ativação de refrigeração (scoolant) e
pré-estágio da ferramenta (pstagetool). A função pstagetool é usada para reduzir o tempo de ciclo. Esta função consiste num processo automático
em que a próxima ferramenta a ser usada no programa é antecipadamente colocada na posição indexada para troca, no armazém robotizado.)
      pbld, n$, sg56, *tlngno$, pfzout, scoolant, pstagetool, e$
      absinc$ = sav_absinc
      pbld, n$, sgabsinc, e$
      pcom_movea
      toolchng = zero
      c_msng$ #Single tool subprogram cal
```

A função **pwcs**, abordada na página anterior, será agora exposta e analisada. Esta função foi alterada para escrever o código **G15** em vez do conjunto de códigos **G54-59**. O código **G15** tem também a particularidade de utilizar um conjunto de registos **Hxx** (não confundir com os registos do *offset* da ferramenta, que também existe no formato **Hxx**), que para uma determinada operação, deve ser escolhido e escrito juntamente com este código modal. Assim sendo, o código alterado utiliza os registos de *offset* especificados nos parâmetros das operações (transmitidos ao pós-processador através da variável **workofs$**, evidenciado a azul), e compara-os com a gama permitida pelo OSP5020M (evidenciado a amarelo), resultado numa mensagem de erro e terminação do pós-processamento, caso o registo **Hxx** esteja fora dessa gama.

```
pwcs  # Configuração do sistemas de eixos do tipo G15
      if mi1$ = one,  (Verificação do valor customizável.)
            [
            sav_frc_wcs = force_wcs
            if sub_level$ > 0, force_wcs = zero
            if workofs$ <> prv_workofs$ | (force_wcs & toolchng),
                  [
                  if ((workofs$>0&workofs$<51) | (workofs$>100&workofs$<151)),  (Aqui verifica-se se Hxx está correto.)
```

```
                [
                h_wcs = workofs$  (Definição de Hxx de acordo com o sistema de coordenadas escolhido na operação.)
                *sg15, *h_wcs  (Aqui escrevem-se a string sg15 (G15) e qual o registo Hxx associado.)
                ]
          else,
"(SISTEMA DE EIXOS G15 DEVE SER : H01-H50 ou H101-H150 ..... POST TERMINADO)", e$
exitpost$  (Aqui chama-se a função de terminação do pós-processamento, quando é detetado um erro nos registos Hxx.)
]
]
force_wcs = sav_frc_wcs
!workofs$
]
```

Relativamente às rotinas de deteção de erros relacionados com a ultrapassagem dos limites dos eixos da máquina, de seguida analisar-se-ão duas rotinas semelhantes, com uma diferença importante. No caso da rotina de deteção dos eixos X e Y (aqui para os efeitos do exemplo apenas se representa o eixo X), a verificação incide apenas na comparação das coordenadas no sistema absoluto. Isto significa que existirá espaço para melhorar a qualidade deste sistema, implementando o mesmo tipo de deteção, mas para as coordenadas em modo de programação relativo (**G91**). A deteção de erros nas coordenadas X e Y funciona da seguinte forma: as diversas posições X e Y que são extraídas no ficheiro NCI, pela função pré-definida **pxyzcout**, são comparadas com os valores dos limites dos eixos, que já se encontram registados. Em caso de ultrapassagem de limites, ativa-se a *flag* de erro correspondente (**errox** ou **erroy**), e chama-se a função de erro (**perrox** ou **perroy**). Esta função tem apenas um intuito: gerar um código de erro que será escrito no ficheiro de código de CN, para cada erro detetado, ou, se o valor customizável **mi4$** for ativado, gerar esse mesmo código de erro e proceder à interrupção do pós-processamento. A função de erro (**perrox** ou **perroy**) compara ainda, num determinado eixo, a posição para a qual foi registado um erro, com a posição seguinte. Em caso de igualdade, é temporariamente desligado o código de erro, pois além de não haver necessidade de repetição, normalmente para esses casos, o código de posição repetido é mesmo omisso.

```
Pxout #X output
     if (xabs>x_limits_max | xabs<x_limits_min), errox = 1  (Aqui é feita a  verificação de ultrapassagem de limites.)
     else, errox = 0
     perrox (Aqui é feita  (Chamada da função de erro, mesmo que não haja erros – neste caso a função apenas emite uma string vazia.)
     if absinc$ = zero,
     [xabs, serrox, !xinc  (A string de erro foi inserida aqui juntamente com o valor da posição X – o mesmo acontece com as outras.)
     perroexit]  (Chamada da função de interrupção do pós-processador.)
```

```
      else, xinc, !xabs
perrox  (Aqui começa a função de erro, que define se a mensagem de erro é escrita juntamente com a posição X, Y ou Z.)
      if errox = 0, serrox = sblank  (Se não há erro, a mensagem é nula.)
      else,
            [result = newfs(two, xabs)
            xabs
            if errox = 1 & xabs<>xabs_b,
(Na linha acima faz –se a comparação entre a posição a ser registada, e a última posição em erro. Se forem iguais, a mensagem de erro é nula.)
                  [serrox = "(<XFDL)"  (Definição da mensagem de erro para o eixo X.)
                  xabs_b = xabs]  (Atualização do valor da última posição X em erro, xabs_b.)
                  else, serrox = sblank
                  ]
```

No caso da deteção de erro com o eixo Z, o processo é semelhante. A única diferença que se evidencia, é a inclusão do cálculo do *offset* da ferramenta.

```
Pzout      #Z output
(Nas linhas abaixo, a lógica do pós-processador segue os mesmos princípios da verificação de erro apresentada na página anterior, para o eixoX.)
      if ((zabs-tool_ol-tool_hl-t_orgin_z$)>z_limits_max |
(zabs+tool_ol+tool_hl+t_orgin_z$)<z_limits_min),
      erroz = 1
      else, erroz = 0
      perroz
      if absinc$ = zero,
      [
      zabs, serroz, !zinc
      perroexit
      ]
      else, zinc, !zabs
```

Finalizando este ponto, analise-se a função de interrupção do pós processador (**perroexit**). Esta função apenas compara a opção escolhida no valor customizável **mi4$**, com a atividade das *flag*s de erro.

```
perroexit
      if mi4$ = 0, sblank
```

```
      if (mi4$ = 1 & (errox = 1 | erroy = 1 | erroz = 1)), serroexit, e$, exitpost$
      else, sblank
```

Alteração aos blocos auxiliares/implementação de novos

Neste ponto faz-se uma análise resumida às alterações efetuadas para que o chamamento de sub-rotinas esteja de acordo com o OSP5020M.

Assim sendo, é possível observar nos excertos abaixo expostos, os seguintes aspetos:

- Adicionou-se uma linha de código que permite à gerir a nomenclatura das sub-rotinas da forma correta: "`osub_prog_n = main_prg_no$`";
- Embora a alteração seja trivial, a mesma só tem sucesso se for aplicada uma formatação à variável "`osub_prog_n`";
  - `fmt  OSUB 4 osub_prog_n`
- Isto faz com que a sub-rotina possua a designação correta, mas com a numeração de acordo com o programa principal;
- Alterou-se também o código "M98" para "CALL", assim como, na função de retorno, "M99" para RTS.

```
psub_call_s$    #Call to sub level
      #result = nwadrs(strosub, sub_prg_no$)
      sub_prg_no$ = sub_prg_no$ + 1000 #Add sub number offset
      if progno$ = sub_prg_no$, result = mprint(sprgnerror)
      osub_prog_n = main_prg_no$
      result = newfs(osub_prog_n, 4)
      pbld, n$, "CALL ", *osub_prog_n, e$

psub_end_s$     #End in sub level
      n$, "RTS", e$
```

ANEXO C : PÓS-PROCESSADOR MODIFICADO, NA ÍNTEGRA

```
[POST_VERSION] #DO NOT MOVE OR ALTER THIS LINE# V14.00 E1 P0 T1232650364 M14.00 I0
# Post Name           : OKUMA MC-40VA v0.9.pst
# Product             : Mill
# Machine Name        : OKUMA MC-40VA
# Control Name        : OSP5020M
# Description         : Pos-processador para o controlador OSP5020M da maquina OKUMA MC-
40VA
# 4-axis/Axis subs.   : No
# 5-axis              : No
# Subprograms         : Yes
# Executable          : MP 14.0
#
# ESTE POS-PROCESSADOR FOI DESENVOLVIDO NO AMBITO DA TESE DE MESTRADO - UTILIZAÇÃO DO
MASTERCAM X5 E OKUMA MC-40VA
# ANTÓNIO LUÍS GALAMBA DE OLIVEIRA FELGUEIRAS DE CARVALHO
# PROFESSOR MANUEL ROMANO BARBOSA
# PROFESSOR PAULO FERREIRA ABREU
#
# Associated File List$
#
# GENERIC FANUC 3X MILL.control-5
#
# Associated File List$
#
# ------------------------------------------------------------------------
# LOG DE REVISÕES DO MASTERCAM (NAO ALTERAR)
# ------------------------------------------------------------------------
# Revision log:
# ------------------------------------------------------------------------
# CNC 06/09/05  -  Initial post setup for Mastercam X
# CNC 10/06/05  -  Changed parameter read for min_speed, modified pspindle, pprep$ and
pset_mach
# CNC 02/03/06  -  Added logic for high-speed toolpath tool inspection (see prapidout &
plinout)
# CNC 06/26/06  -  Initial post setup for Mastercam X2
# CNC 02/26/07  -  Modified pwcs
# CNC 11/02/07  -  Added prv_shftdrl$ = zero
# CNC 04/08/08  -  X3 release - Removed check for write_ops
# CNC 01/26/09  -  Initial post update for Mastercam X4
# CNC 06/09/09  -  Updated MD parameters
# CNC 02/03/10  -  Initial post update for Mastercam X5
# CNC 08/17/10  -  Added fix for canned drill cycle incremental mode code output and Z
output in incremental mode
#               -  Added fix for X coolant output
#               -  Added fix for stock to leave output in tool table
#               -  Removed CD_VAR variables
# ------------------------------------------------------------------------
# FEATURES DO POS-PROCESSADOR, AUTORIA DA MASTERCAM (NAO ALTERAR)
# ------------------------------------------------------------------------
# Features:
# ------------------------------------------------------------------------
# This post supports Generic Fanuc code for 3 axis milling.
# It is designed to support the features of Mastercam X Mill.
#
# NEW FEATURES FOR X:
# - Sub-program support
# - Machine definition, control definition and toolpath group parameter read sections
added.
# - Post sets min/max spindle speed and max feed rate from MD and CD.
# - Variable initialization with SET_BY_MD or SET_BY_CD are overwritten in this post by
parameter or
#     variable settings from MD or CD.
# - Enhanced tool information - Added switch for tool comments only, tooltable in header
with no tool
#      comments at tool change or tooltable in header with tool comments at tool change
(see tool_info)
```

```
#     Tooltable output includes cutter compensation type and stock to leave information
# - Enhanced tool staging options - enable or disable in CD.  Set stagetltype in post
for output type:
#     Do not stage 1st tool, stage 1st tool at last tool change or stage 1st tool at end
of file (peof)
# - Supports X comments including machine name, group name and group comment output (see
pcomment2)
# - Additional date, time and data path output options (see pheader)
# - Additional rigid tapping cycle (separate from original tapping cycle) and initial
custom drill
#     cycle support (see pmisc2$ and pdrlcst$)
# - Support for 10 additional canned text options for X
# - Decimal support for sequence number output (set "Increment sequence number" in CD to
a decimal value
#     for output.  I.E. "Increment sequence number" = .5, "Start sequence number" = 10 :
N10, N10.5, N11, N11.5, etc...)
# - Switch for output of M00 or M01 at tool change (3 position switch, off, M00, M01 -
see prog_stop)
# - Support for seperate XY, XZ and YZ plane/arc variables (see Arc page in CD)
# - Support for X style coolant.  Allows up to 10 different coolants to be turned on/off
before, with, or after like
#     canned text.  Coolant output is handled by "coolant" variable and string selector
for V9 style coolant,
#     "coolantx" variable and string selector for X style coolant.
#
# ------------------------------------------------------------------------
# Valores customizáveis:
# ------------------------------------------------------------------------
# Inteiros:
#
# mi1 - Modo de sistema de sistemas de coordenadas
#        1 = Usar G15: H01-H50 | H101-H150 -> com rotina de erro
#        2 = Usar G92
#
# mi2 - Programação em modo absoluto ou incremental
#        0 = absoluto
#        1 = incremental
#
# mi3 - Ponto casa (de retorno) definido por G28 ou G30
#        0 = G28
#        1 = G30
#
#  mi4  -  Erros  de  saída  de  limites  interrompem  post  (provoca  a  paragem  do  pós
processamento em caso de erro)
#        0 = Não
#        1 = Sim
#
# mi5 - Reservado
#
#
#  mi6  -  Função  VAPA*  -  Value  Of  Actual  Position  of  Axis  X  Y  Z  (EXPERIMENTAL,  A
DESENVOLVER, inibido no POSPROC.)
#       0 = Não
#       1 = Sim
#
# mi7 - mi10 (NOT USED)
#
# Reals:
#
# mr1 - mr10 (NOT USED)
# ------------------------------------------------------------------------
# Informações fornecidas de origem - NÃO APAGAR
# ------------------------------------------------------------------------
# ------------------------------------------------------------------------
#Canned text:
#    Entering cantext on a contour point from within Mastercam allows the
#    following functions to enable/disable.
#    Cantext value:
#    1 = Program Stop = output the "M00" stop code
#    2 = Optional Stop =  output the "M01" optional stop code
#    3 = Block Delete on = turn on block delete codes in NC lines
#    4 = Block Delete off = turn off block delete codes in NC lines
#
# ------------------------------------------------------------------------
#Additional Notes:
# 1) G54 calls are generated where the work offset entry of 0 = G54,
```

```
#    1 = G55, etc.
# 2) Metric is applied from the NCI met_tool variable.
# 3) Incremental mode calculates motion from home position at toolchanges.
#    The home position is used to define the last position of the tool
#    for all toolchanges.
# 4) The variable 'absinc' is now pre-defined, set mi2 (Misc. Integer) for
#    the 'top level' absolute/incremental program output.  Subprograms are
#    updated through the Mastercam dialog settings for sub-programs.
# 5) Transform subprograms are intended for use with G54.. workshifts.
#
# END_HEADER$
#
# ----------------------------------------------------------------------
# Debugging and Factory Set Program Switches
# ----------------------------------------------------------------------
#Define Constants
m_one        := -1
zero         := 0
one          := 1
two          := 2
three        := 3
four         := 4
five         := 5
c9k          := 9999

bug4$        : 1     #Debug output with the tilde '~'.
                     #A value greater the zero applies the variable formatting with
                     #debug output (default is typically FS 1 but not a guarantee).
                     #A value of zero gets the value directly with NO formatting.

linktolvar$  : 0     #Associate X tolerance variables to V9- variable?
linkplnvar$  : 0     #Associate X plane specific variables to V9- variable?

skp_lead_flgs$ : 0   #Do NOT use v9 style contour flags
get_1004$    : 1     #Find gcode 1004 with getnextop?
rpd_typ_v7$  : 0     #Use Version 7 style contour flags/processing?
strtool_v7$  : 2     #Use Version 7+ toolname?
tlchng_aft$  : 2     #Delay call to toolchange until move line
cant_tlchng$ : 1     #Ignore cantext entry on move with tlchng_aft
newglobal$   : 1     #Error checking for global variables
getnextop$   : 1     #Build the next variable table
tooltable$   : 3     #Pre-read, call the pwrtt postblock


# ----------------------------------------------------------------------
# General Output Settings
# ----------------------------------------------------------------------
maxfeedpm    : 500   #SET_BY_MD Limit for feed in inch/min
ltol_m       : 0.05  #Length tolerance for arccheck, metric
vtol_m       : 0.0025#System tolerance, metric
maxfeedpm_m  : 10000 #SET_BY_MD Limit for feed in mm/min
force_wcs    : yes$  #Force WCS output at every toolchange?
stagetool    : 1     #SET_BY_CD 0 = Do not pre-stage tools, 1 = Stage tools
stagetltype  : 1     #0 = Do not stage 1st tool
                     #1 = Stage 1st tool at last tool change
                     #2 = Stage 1st tool at end of file (peof)
use_gear     : 0     #Output gear selection code, 0=no, 1=yes
min_speed    : 50    #SET_BY_MD Minimum spindle speed
progname$    : 1     #Use uppercase for program name (sprogname)
prog_stop    : 1     #Program stop at toolchange: 0=None, 1=M01, 2 = M00
tool_info    : 2     #Output tooltable information?
                     #0 = Off - Do not output any tool comments or tooltable
                     #1 = Tool comments only
                     #2 = Tooltable in header - no tool comments at T/C
                     #3 = Tooltable in header - with tool comments at T/C
tlchg_home   : no$   #Zero return X and Y axis prior to tool change?

# ----------------------------------------------------------------------
# Common User-defined Variable Initializations (not switches!)
# ----------------------------------------------------------------------
xia          : 0     #Formatted absolute value for X incremental calculations
yia          : 0     #Formatted absolute value for Y incremental calculations
zia          : 0     #Formatted absolute value for Z incremental calculations

bld          : 0     #Block delete active
result       : 0     #Return value for functions
```

```
sav_spc      : 0     #Save spaces
sav_gcode    : 0     #Gcode saved
sav_absinc   : 0     #Absolute/Incremental Saved Value
sav_coolant  : 0     #Coolant saved
sav_frc_wcs  : 0     #Force work offset flag saved
toolchng     : 1     #On a toolchange flag
spdir2       : 1     #Copy for safe spindle direction calculation

#Drill variables
drlgsel      : -1    #Drill Select Initialize
drillref     : 0     #Select drill reference
drlgcode     : 0     #Save Gcode in drill
sav_dgcode   : 0     #Drill gcode saved

#Subprogram variables
mr_rt_actv   : 0     #Flag to indicate if G51/G68 is active
                     #0=Off, 1=Toolchange, 2=Subprogram call/start, G68
                     #3=Absolute start, both
rt_csav      : 0     #C saved value
end_sub_mny  : 0     #Many tool setting captured at transform sub end
comp_type      : 0        #Cutter  compensation  type  -  0=computer,  1=control,  2=wear,
3=reverse wear, 4=off
#Coolant variables for X style coolant
cant_pos     : 0     #Read from current canned text (cant_pos1 - cant_pos20)
coolant_bin  : 0     #Binary value for current coolant command
coolant_on   : 0     #Binary value holding the sum of all coolants currently on
coolantx     : 0     #Selector variable for coolant string selector
local_int    : 0     #Local variable for output of coolant off commands
result2      : 0     #Return value for functions
suppress     : 0     #Flag used to suppress redundant coolant on commands
all_cool_off : 0     #First coolant off command shuts off ALL coolant options

#Variables to capture parameter values - use to set post switches in pset_mach
rotaxerror    : 0      #Error flag - Output error message if rotary axis is detected in
active axis combination
rot_ax_cnt   : 0     #Rotary axis counter
component_type : 0   #Component type: (See documentation for complete list - )
                       #0 = MACHINE
                       #1 = STOCK_COMPONENT
                       #2 = MISC_COMPONENT
                       #3 = MACHINE_BASE_COMPONENT
                       #4 = LINEAR_AXIS_COMPONENT
                       #5 = ROTARY_AXIS_COMPONENT
                       #6 = RECT_TABLE_COMPONENT
                       #12 = CHUCK_COMPONENT
                       #24 = TOOL_SPINDLE_COMPONENT
                       #23 = ATC_COMPONENT
ra_error     : 0     #Flag for aggregate operation

# ------------------------------------------------------------------------
#variáveis e strings específicos deste pós-processador
# ------------------------------------------------------------------------
x_init       : 0 # posição inicial x (opcional)
y_init       : 0 # posição inicial y (opcional)
z_init       : 0 # posição inicial z (opcional)
x_limits_min : 0 # limite x inferior
x_limits_max : 0 # limite x superior
errox : 0        # flag de erro
y_limits_min : 0 # limite y inferior
y_limits_max : 0 # limite y superior
erroy : 0        # flag de erro
z_limits_min : 0 # limite z inferior
z_limits_max : 0 # limite z superior
erroz : 0        # flag de erro
mi4$ : 0         # opção de terminar pós-processador durante verificação de limites
mi5$ : 0         # RESERVADO
mi6$ : 0         # VAPA* (EXPERIMENTAL)
xabs_b : 0     # valor absoluto anterior de X
yabs_b : 0     # valor absoluto anterior de Y
zabs_b : 0     # valor absoluto anterior de Z
comp_n2: 0     # ID do componente (para efeitos de debugging)
comp_n : 0     # ID do componente
comp_i : 0     # ID do componente inicial no NCI
comp_f : 0     # ID do componente final no NCI
ent_i  : 0     # entidade inicial para scan de variáveis
saxis_lbl : "" # label do eixo
```

```
strx    : "X"  # string que representa X
stry    : "Y"  # string que representa Y
strz    : "Z"  # string que representa Z
codestart : 0  #
x_set : 0      #flag usada na procura de dados do eixo X
y_set : 0      #flag usada na procura de dados do eixo Y
z_set : 0      #flag usada na procura de dados do eixo Z
osub_prog_n  : 0 #designação do programa da subrotina
tool_ol      :0 #tool overall length - comprimento total da ferramenta sem fixação
tool_hl      :0 #tool holder length - comprimento da fixação
tool_tl      :0 #tool total length = tool_ol+tool_hl
tool_d       :0 #tool diâmeter - diâmetro da ferramenta
fto          :0 #first tool offset - offset da primeira ferramenta
sfilename    : "" #string do nome do ficheiro
#conjunto de mensagens de erro de programação fora de limites
serrox       : ""
serrox_y     : "(XFDL - X FORA DE LIMITES)"
serroy       : ""
serroy_y     : "(YFDL - Y FORA DE LIMITES)"
serroz       : ""
serroz_y     : "(ZFDL - Z FORA DE LIMITES)"
serroexit    : "(VERIFICAR LIMITES DO PROGRAMA : POST TERMINADO)"



# ------------------------------------------------------------------------
#String and string selector definitions for NC output
# ------------------------------------------------------------------------
#Address string definitions
strm         : "M"
strn         : "N"
stro         : "O"
strosub      : "OSUB" #chamamento de subrotina
strrts       : "RTS"  #retorno de subrotina
srad         : "R"
srminus      : "R-"
sblank       : ""

#Cantext string definitions (spaces must be padded here)
sm00         : "M00"
sm01         : "M01"
strtextno    : ""
strcantext   : ""

#Transform mirror and rotate codes
strns_mir_on  : "G51.1" #Programmable mirror image code
strns_mir_off : "G50.1" #Programmable mirror image cancel code
strns_rot_on  : "G68"   #Coordinate System Rotation
strns_rot_off : "G69"   #Coordinate System Rotation Cancel

#Misc. string definitions
sopen_prn    : "("   #String for open parenthesis "("
sclose_prn   : ")"   #String for close parenthesis ")"
sdelimiter   : "|"   #String for delimiter
sg94         : "G94" #Units per minute
sg95         : "G95" #Feed per rotation
sm29         : "M29" #Rigid tapping preperation support function
sg80         : "G80" #Cancel canned drilling cycle
sg43         : "G43" #Cancelamento compensação tridimensional
sg49         : "G49" #Cancelamento do offset da ferramenta (FANUC - não usado)
sg92         : "G92" #Sistema de eixos da peça
sm06         : "M6"  #Toolchange
sg15         : "G15" #Sistema de eixos do tipo H1-50, H101-150
sg56           : "G56" #Compensação do comprimento da ferramenta (sentido Z, usado para
offset de trabalho)

# ------------------------------------------------------------------------
# Error messages
# ------------------------------------------------------------------------
shomeserror  : "ERROR - WORK OFFSET USAGE DOES NOT SUPPORT TRANSFORM SUBPROGRAM"
sprgnerror   : "ERROR - SUBPROGRAM NUMBER MATCHES THE MAIN PROGRAM NUMBER"
sra_error    : "ERROR - AGGREGATES NOT SUPPORTED IN THIS POST"
srotaxerror  : "ERROR - ROTARY AXIS DETECTED IN SELECTED AXIS COMBINATION- POST DOES NOT
SUPPORT ROTARY AXIS OUTPUT"
```

```
# ------------------------------------------------------------------------
# General G and M Code String select tables
# ------------------------------------------------------------------------
# Motion G code selection
sg00    : "G0"       #Rapid
sg01    : "G1"       #Linear feed
sg02    : "G2"       #Circular interpolation CW
sg03    : "G3"       #Circular interpolation CCW
sg04    : "G4"       #Dwell
sgcode  : ""         #Target string

fstrsel sg00 gcode$ sgcode 5 -1
# ------------------------------------------------------------------------
# Select work plane G code
sg17    : "G17"      #XY plane code
sg19    : "G19"      #YZ plane code
sg18    : "G18"      #XZ plane code
sgplane : ""         #Target string

fstrsel sg17 plane$ sgplane 3 -1
# ------------------------------------------------------------------------
#Select english/metric code
sg20    : "G20"      #Inch code
sg21    : "G21"      #Metric code
smetric : ""         #Target string

fstrsel sg20 met_tool$ smetric 2 -1
# ------------------------------------------------------------------------
#Select reference return code
sg28    : "G28"      #First reference point return
sg30    : "G30 P0"   #Retorno à posição CASA, ponto P0 = posição neutra dos eixos
sg28ref : ""         #Target string

fstrsel sg28 mi3$ sg28ref 2 -1
# ------------------------------------------------------------------------
# Cutter compensation G code selection
scc0    : "G40"      #Cancel cutter compensation
scc1    : "G41"      #Cutter compensation left
scc2    : "G42"      #Cutter compensation right
sccomp  : ""         #Target string

fstrsel scc0 cc_pos$ sccomp 3 -1
# ------------------------------------------------------------------------
# Canned drill cycle string select
sg81    : "G81"      #drill - no dwell
sg81d   : "G82"      #drill - with dwell
sg83    : "G83"      #peck drill - no dwell
sg83d   : "G83"      #peck drill - with dwell
sg73    : "G73"      #chip break - no dwell
sg73d   : "G73"      #chip break - with dwell
sg84    : "G84"      #tap - right hand
sg84d   : "G74"      #tap - left hand
sg85    : "G85"      #bore #1 - no dwell
sg85d   : "G89"      #bore #1 - with dwell
sg86    : "G86"      #bore #2 - no dwell
sg86d   : "G86"      #bore #2 - with dwell
sgm1    : "G76"      #fine bore - no dwell
sgm1d   : "G76"      #fine bore - with dwell
sgm2    : "G84"      #rigid tap  - right hand
sgm2d   : "G74"      #rigid tap  - left hand
sgdrill : ""         #Target string

fstrsel sg81 drlgsel sgdrill 16 -1
# ------------------------------------------------------------------------
# Select incremental or absolute G code
sg90    : "G90"      #Absolute code
sg91    : "G91"      #Incremental code
sgabsinc : ""        #Target string

fstrsel sg90 absinc$ sgabsinc 2 -1
# ------------------------------------------------------------------------
#Canned drill cycle reference height
sg98    : "G98"      #Reference at initht
sg99    : "G71"      #Posição de retorno de ciclos de furação (No FANUC era G99)
sgdrlref : ""        #Target string
```

```
fstrsel sg98 drillref sgdrlref 2 -1
# ------------------------------------------------------------------------
# Generate string for spindle
sm04    : "M4"       #Spindle reverse
sm05    : "M5"       #Spindle off
sm03    : "M3"       #Spindle forward
spindle : ""         #Target string

fstrsel sm04 spdir2 spindle 3 -1
# ------------------------------------------------------------------------
# Coolant M code selection for V9 style coolant
# Note: To enable V9 style coolant, click on the General Machine Parameters icon
#   in the Machine Definition Manager, Coolant tab, enable first check box
#   Output of V9 style coolant commands in this post is controlled by scoolant
sm09    : "M9"       #Coolant Off
sm08    : "M8"       #Coolant Flood
sm08_1  : "M7"       #Coolant Mist
sm08_2  : "M51"      #Refrigeração através da ferramenta, alta pressão
scoolant : ""        #Target string

fstrsel sm09 coolant$ scoolant 4 -1
# ------------------------------------------------------------------------
# Coolant output code selection for X style coolant
# Note: To enable X style coolant, click on the General Machine Parameters icon
#   in the Machine Definition Manager, Coolant tab, disable first check box
#    Output of X style coolant commands in this post is controlled by pcan, pcan1, &
pcan2
scool50 : "M8"                #Coolant 1 on value
scool51 : "M9"                #Coolant 1 off value
scool52 : "M7"                #Coolant 2 on value
scool53 : "M9"                #Coolant 2 off value
scool54 : "M88"               #Coolant 3 on value
scool55 : "M89"               #Coolant 3 off value
scool56 : "M8(Coolant4=ON)"   #Coolant 4 on value
scool57 : "M9(Coolant4=OFF)"  #Coolant 4 off value
scool58 : "M8(Coolant5=ON)"   #Coolant 5 on value
scool59 : "M9(Coolant5=OFF)"  #Coolant 5 off value
scool60 : "M8(Coolant6=ON)"   #Coolant 6 on value
scool61 : "M9(Coolant6=OFF)"  #Coolant 6 off value
scool62 : "M8(Coolant7=ON)"   #Coolant 7 on value
scool63 : "M9(Coolant7=OFF)"  #Coolant 7 off value
scool64 : "M8(Coolant8=ON)"   #Coolant 8 on value
scool65 : "M9(Coolant8=OFF)"  #Coolant 8 off value
scool66 : "M8(Coolant9=ON)"   #Coolant 9 on value
scool67 : "M9(Coolant9=OFF)"  #Coolant 9 off value
scool68 : "M8(Coolant10=ON)"  #Coolant 10 on value
scool69 : "M9(Coolant10=OFF)" #Coolant 10 off value
scoolantx : ""                #Target string

fstrsel scool50 coolantx scoolantx 20 -1
# ------------------------------------------------------------------------
#X coolant has the option - First coolant off command shuts off ALL coolant options
sall_cool_off : "M09" #Coolant off command output with all_cool_off

# ------------------------------------------------------------------------
# Define the gear selection code
flktbl  1       3       #Lookup table definition - table no. - no. entries
        40      0       #Low gear range
        41      400     #Med gear range
        42      2250    #Hi gear range

# ------------------------------------------------------------------------
# Define coolant binary value for X style coolant
flktbl  2       20      #Lookup table definition - table no. - no. entries
        1       50      #Coolant 1 on value
        2       51      #Coolant 1 off value
        4       52      #Coolant 2 on value
        8       53      #Coolant 2 off value
        16      54      #Coolant 3 on value
        32      55      #Coolant 3 off value
        64      56      #Coolant 4 on value
        128     57      #Coolant 4 off value
        256     58      #Coolant 5 on value
        512     59      #Coolant 5 off value
        1024    60      #Coolant 6 on value
```

```
        2048    61      #Coolant 6 off value
        4096    62      #Coolant 7 on value
        8192    63      #Coolant 7 off value
        16384   64      #Coolant 8 on value
        32768   65      #Coolant 8 off value
        65536   66      #Coolant 9 on value
        131072  67      #Coolant 9 off value
        262144  68      #Coolant 10 on value
        524288  69      #Coolant 10 off value

# ------------------------------------------------------------------------
# Month selector
smon0   : ""
smon1   : "JAN."
smon2   : "FEV."
smon3   : "MAR."
smon4   : "ABR."
smon5   : "MAI."
smon6   : "JUN."
smon7   : "JUL."
smon8   : "AGO."
smon9   : "SET."
smon10  : "OUT."
smon11  : "NOV."
smon12  : "DEC."
smonth  : ""         #Target string

fstrsel smon0 month$ smonth 13 -1
# ------------------------------------------------------------------------
# Cutter Compensation Type
scomp   : "COMPUTER"
scomp1  : "CONTROL COMP"
scomp2  : "WEAR COMP"
scomp3  : "REVERSE WEAR COMP"
scomp4  : "OFF"
scomp_type : ""      #Target string

fstrsel scomp comp_type scomp_type 5 -1
# ------------------------------------------------------------------------
# Format statements - n=nonmodal, l=leading, t=trailing, i=inc, d=delta
# ------------------------------------------------------------------------
#Default english/metric position format statements
fs2 1   0.7 0.6      #Decimal, absolute, 7 place, default for initialize (:)
fs2 2   0.4 0.3      #Decimal, absolute, 4/3 place
fs2 3   0.4 0.3d     #Decimal, delta, 4/3 place
#Common format statements
fs2 4   1 0 1 0      #Integer, not leading
fs2 5   2 0 2 0l     #Integer, force two leading
fs2 6   3 0 3 0l     #Integer, force three leading
fs2 7   4 0 4 0l     #Integer, force four leading
fs2 9   0.1 0.1      #Decimal, absolute, 1 place
fs2 10  0.2 0.2      #Decimal, absolute, 2 place
fs2 11  0.3 0.3      #Decimal, absolute, 3 place
fs2 12  0.4 0.4      #Decimal, absolute, 4 place
fs2 13  0.5 0.5      #Decimal, absolute, 5 place
fs2 14  0.3 0.3d     #Decimal, delta, 3 place
fs2 15  0.2 0.1      #Decimal, absolute, 2/1 place (feedrate)
fs2 16  1 0 1 0n     #Integer, forced output
fs2 17  0.2 0.3      #Decimal, absolute, 2/3 place (tapping feedrate)

# These formats used for 'Date' & 'Time'
fs2 18  2.2 2.2lt    #Decimal, force two leading & two trailing (time2)
fs2 19  2 0 2 0t     #Integer, force trailing                   (hour)
fs2 20  0 2 0 2lt    #Integer, force leading & trailing         (min)

# This format statement is used for sequence number output
# Number of places output is determined by value for "Increment Sequence Number" in CD
# Max depth to the right of the decimal point is set in the fs statement below
fs2 21  0^7 0^7      #Decimal, 7 place, omit decimal if integer value

# ------------------------------------------------------------------------
# Toolchange / NC output Variable Formats
# ------------------------------------------------------------------------
fmt  T  4   t$          #Tool number
fmt  T  4   first_tool$ #First tool used
fmt  T  4   next_tool$  #Next tool used
```

```
fmt  D  4   tloffno$    #Diameter offset number
fmt  H  4   tlngno$     #Length offset number
fmt  G  4   g_wcs       #WCS G address
fmt  P  4   p_wcs       #WCS P address
fmt  H  5   h_wcs
fmt  S  4   speed       #Spindle Speed
fmt  M  4   gear        #Gear range
# ------------------------------------------------------------------------
fmt  N  21  n$          #Sequence number
fmt  X  2   xabs        #X position output
fmt  Y  2   yabs        #Y position output
fmt  Z  2   zabs        #Z position output
fmt  X  3   xinc        #X position output
fmt  Y  3   yinc        #Y position output
fmt  Z  3   zinc        #Z position output
fmt  I  3   iout        #Arc center description in X
fmt  J  3   jout        #Arc center description in Y
fmt  K  3   kout        #Arc center description in Z
fmt  R  2   arcrad$     #Arc Radius
fmt  F  15  feed        #Feedrate
fmt  P  11  dwell$      #Dwell
fmt  M  5   cantext$    #Canned text
fmt  F  2   pitch       #Tap pitch (units per thread)
# ------------------------------------------------------------------------
#Move comment (pound) to output colon with program numbers
fmt  O  7   progno$     #Program number
#fmt ":" 7   progno$     #Program number
fmt  O  7   main_prg_no$ #Program number
#fmt ":" 7   main_prg_no$ #Program number
fmt  O  7   sub_prg_no$ #Program number
#fmt ":" 7   sub_prg_no$ #Program number
fmt  X  2   sub_trnsx$  #Rotation point
fmt  Y  2   sub_trnsy$  #Rotation point
fmt  Z  2   sub_trnsz$  #Rotation point
fmt  OSUB 4 osub_prog_n #formato para subrotina
# ------------------------------------------------------------------------
fmt  Q  2   peck1$      #First peck increment (positive)
fmt  Q  2   shftdrl$    #Fine bore tool shift
fmt  R  2   refht_a     #Reference height
fmt  R  2   refht_i     #Reference height
# ------------------------------------------------------------------------
fmt "TOOL - "      4   tnote    #Note format
fmt "DIA. OFF. - " 4   toffnote #Note format
fmt "LEN. - "      4   tlngnote #Note format
fmt "TOOL DIA. - " 1   tldia$   #Note format
fmt "XY STOCK TO LEAVE - " 2  xy_stock #Note format
fmt "Z STOCK TO LEAVE - "  2  z_stock  #Note format
#formatos para as variáveis implementadas
fmt 4 x_limits_min
fmt 4 x_limits_max
fmt 4 y_limits_min
fmt 4 y_limits_max
fmt 4 z_limits_min
fmt 4 z_limits_max
fmt 2 xabs_b
fmt 2 yabs_b
fmt 2 zabs_b
fmt 2 tool_ol
fmt 2 tool_hl
fmt 2 tool_tl
fmt 11 x_init
fmt 11 y_init
fmt 11 z_init
# ------------------------------------------------------------------------
fmt     4   year2       #Calculated year value
fmt     18  time2       #Capture 24-hour time value into 'time2' variable
fmt     19  hour        #Hour
fmt     20  min         #Minutes
year2 = year$ + 2000

# ------------------------------------------------------------------------
# Tool Comment / Manual Entry Section
# ------------------------------------------------------------------------
ptoolcomment    #Comment for tool
      tnote = t$, toffnote = tloffno$, tlngnote = tlngno$
      if tool_info = 1 | tool_info = 3,
```

```
        sopen_prn,  pstrtool,  sdelimiter,  *tnote,  sdelimiter,  *toffnote,  sdelimiter,
*tlngnote, sdelimiter, *tldia$, sclose_prn, e$

ptooltable      #Tooltable output
      tool_tl = tool_ol+tool_hl
      sopen_prn, *t$, sdelimiter, pstrtool, sdelimiter, *tlngno$,
        [if  comp_type  >  0  &  comp_type  <  4,  sdelimiter,  *tloffno$,  sdelimiter,
*scomp_type, sdelimiter, *tldia$],
        [if xy_stock <> 0 | z_stock <> 0, sdelimiter, *xy_stock, sdelimiter, *z_stock],
        sclose_prn, e$
      xy_stock = 0  #Reset stock to leave values
      z_stock = 0   #Reset stock to leave values

pstrtool        #Comment for tool
      if strtool$ <> sblank,
        [
        strtool$ = ucase(strtool$)
        *strtool$
        ]

pcomment$       #Comment from manual entry (must call pcomment2)
      pcomment2 #Required if doing boolean 'if' logic testing!

pcomment2       #Output Comment from manual entry
      scomm$ = ucase (scomm$)
      if gcode$ = 1005, sopen_prn, scomm$, sclose_prn, e$  #Manual entry - as comment
      if gcode$ = 1006, scomm$, e$            #Manual entry - as code
      if gcode$ = 1007, sopen_prn, scomm$, sclose_prn       #Manual entry - as comment
with move NO e$
      if gcode$ = 1026, scomm$                #Manual entry - as code with move NO e$
      if gcode$ = 1008, sopen_prn, scomm$, sclose_prn, e$  #Operation comment
      if gcode$ = 1051, sopen_prn, scomm$, sclose_prn, e$  #Machine name
      if gcode$ = 1052, sopen_prn, scomm$, sclose_prn, e$  #Group comment
      if gcode$ = 1053, sopen_prn, scomm$, sclose_prn, e$  #Group name
      if gcode$ = 1054, sopen_prn, scomm$, sclose_prn, e$  #File Descriptor

# ------------------------------------------------------------------------
# Start of File and Toolchange Setup
# ------------------------------------------------------------------------
plimits
      #if ent_i>0,
      [rd_mch_ent_no$ = ent_i
       rd_md$]

ptime           #Convert 24-hour time format into 12-hour AM/PM format
      if time$ >= 13, time2 = (time$ - 12)
      else, time2 = time$
      hour = int(time2), min = frac(time2)
      *hour, ":", *min,
      if time$ > 12, " PM"
      else, " AM"

pheader$         #Call before start of file

      sfilename = "$" + sprogname$ + ".MIN" + "%"
      sfilename, e$
      sav_spc = spaces$
      spaces$ = 0
      *progno$, sopen_prn, sprogname$, sclose_prn, e$
      #sopen_prn, "PROGRAM NAME - ", sprogname$, sclose_prn, e$
      sopen_prn,  "DATE=DD-MM-YY  - ",  date$,  "  TIME=HH:MM  -  ",  time$,  sclose_prn,  e$
#Date and time output Ex. 12-02-05 15:52
      #sopen_prn, "DATE - ", month$, "-", day$, "-", year$, sclose_prn, e$  #Date output
as month,day,year - Ex. 02-12-05
      #sopen_prn,  "DATE  -  ",  *smonth,  "  ",  day$,  "  ",  *year2,  sclose_prn,  e$  #Date
output as month,day,year - Ex. Feb. 12 2005
      #sopen_prn, "TIME - ", time$, sclose_prn, e$  #24 hour time output - Ex. 15:52
      #sopen_prn, "TIME - ", ptime sclose_prn, e$  #12 hour time output 3:52 PM
      spathnc$ = ucase(spathnc$)
      smcname$ = ucase(smcname$)
      stck_matl$ = ucase(stck_matl$)
      snamenc$ = ucase(snamenc$)
      sopen_prn, "MCX FILE - ", *smcpath$, *smcname$, *smcext$, sclose_prn, e$
      sopen_prn, "NC FILE - ", *spathnc$, *snamenc$, *sextnc$, sclose_prn, e$
      sopen_prn, "MATERIAL - ", *stck_matl$, sclose_prn, e$
      sopen_prn, "LIMITES - CODIGOS DE ERRO:", sclose_prn, e$
```

```
      *serrox_y, e$
      *serroy_y, e$
      *serroz_y, e$
      spaces$ = sav_spc
      codestart=1
      plimits
      sopen_prn, "LIMITES DOS EIXOS :", sclose_prn, e$
      sopen_prn, "X ["*x_limits_min, ";" *x_limits_max, "]"sclose_prn, e$
      sopen_prn, "Y ["*y_limits_min, ";" *y_limits_max, "]"sclose_prn, e$
      sopen_prn, "Z ["*z_limits_min, ";" *z_limits_max, "]"sclose_prn, e$
      #*stck_wdth$,  *stck_ht$,  *stck_thck$,  *stck_x$,  *stck_y$,  *stck_z$,  *stck_crnr$,
e$
      #*tool_d, e$
      #pposstockcalc
      #*x_init, *y_init, *z_init, e$


#pposstockcalc
#      if mi5 = 1,
#      [
#       x_init = stck_x$ - stck_ht$/2 - tool_d/2
#       y_init = stck_y$ - stck_wdth$/2 - tool_d/2
#       z_init = z_limits_max
#        #z_init = stck_z$ + tool_d
#      ]


psof0$           #Start of file for tool zero
      psof$

psof$            #Start of file for non-zero tool number


      if ra_error = 1, [if mprint(sra_error, 2) = 2, exitpost$]
      toolchng = one
      if ntools$ = one,
        [
        #skip single tool outputs, stagetool must be on
        stagetool = m_one
        !next_tool$
        ]
      pbld, n$, *smetric, e$
      pbld, n$, *sgcode, *sgplane, scc0, "G53", sg80, *sgabsinc, e$
      sav_absinc = absinc$
      if mi1$ = 2, #Work coordinate system
        [
        absinc$ = one
        #pfbld, n$, sgabsinc, *sg28ref, "Z0.", e$
        #pfbld, n$, *sg28ref, "X0.", "Y0.", e$
        if mi6$ = 1,
        [
        pbld, n$, sg92, "VAPAX", "VAPAY", zh$, e$
        ]
        if mi6$ = 0,
        [
        pbld, n$, sg92, *xh$, *yh$, *zh$, e$
        ]
        absinc$ = sav_absinc
        ]
      pcom_moveb
      pcheckaxis
      c_mmlt$ #Multiple tool subprogram call
      ptoolcomment
      comment$
      pcan
      pbld, n$, *t$, sm06, e$
      if mi1$ = one, absinc$ = zero
      pcan1, pbld, n$,  pwcs, e$
      pbld, n$, *sgcode, *sgabsinc, pfxout, pfyout,
        [if nextdc$ <> 7, *speed, *spindle], pgear, strcantext, e$
      pbld, n$, sg56, *tlngno$, pfzout, scoolant, pstagetool, e$
      absinc$ = sav_absinc
      pbld, n$, sgabsinc, e$
      pcom_movea
      toolchng = zero
      c_msng$ #Single tool subprogram call
```

```
ptlchg0$         #Call from NCI null tool change (tool number repeats)

      pcom_moveb
      pcheckaxis
      c_mmlt$ #Multiple tool subprogram call
      comment$
      pcan
      result = newfs(15, feed)  #Reset the output format for 'feed'
      pbld, n$, sgplane, e$
      pspindchng
      pbld, n$, scoolant, e$
      if mi1$ = one & workofs$ <> prv_workofs$,
        [
        sav_absinc = absinc$
        absinc$ = zero
        pbld, n$, sgabsinc, pwcs, e$
        pbld, n$, pfxout, pfyout, pfzout, e$
        pe_inc_calc
        ps_inc_calc
        absinc$ = sav_absinc
        ]
      if gcode$ = one, plinout
      else, prapidout
      pcom_movea
      c_msng$ #Single tool subprogram call

ptlchg$          #Tool change
      toolchng = one
      if mi1$ = 2, #Work coordinate system
        [
        #pfbld, n$, *sg28ref, "X0.", "Y0.", e$
        #pbld, n$, sg92, *xh$, *yh$, *zh$, e$
        ]
      if prog_stop = 1, pbld, n$, *sm01, e$
      if prog_stop = 2, pbld, n$, *sm00, e$
      pcom_moveb
      pcheckaxis
      c_mmlt$ #Multiple tool subprogram call
      ptoolcomment
      comment$
      pcan
      result = newfs(15, feed)  #Reset the output format for 'feed'
      pbld, n$, *t$, sm06, e$
      sav_absinc = absinc$
      if mi1$ = one, absinc$ = zero
      pcan1, pbld, n$, *sgcode, !sgabsinc, pwcs, e$
      pbld, n$, pfxout, pfyout,
        [if nextdc$ <> 7, *speed, *spindle], pgear, strcantext, e$
      pbld, n$, sg56, *tlngno$, pfzout, scoolant, pstagetool, e$
      absinc$ = sav_absinc
      pbld, n$, sgabsinc, e$
      pcom_movea
      toolchng = zero
      c_msng$ #Single tool subprogram call

pstagetool      #Pre-stage tools
      if stagetool = 1,
        [
        if ttblend$,  #Check for last toolchange
          [
          if stageltltype = 1, *next_tool$  #stage first tool at last toolchange
          ]
        else, *next_tool$ #stage tool at every toolchange
        ]

pretract        #End of tool path, toolchange
      sav_absinc = absinc$
      absinc$ = one
      sav_coolant = coolant$
      coolant$ = zero

#      if nextop$ = 1003, #Uncomment this line to leave coolant on until eof unless
        [                 #  explicitely turned off through a canned text edit
        if all_cool_off,
          [
```

```
          #all coolant off with a single off code here
          if coolant_on, pbld, n$, sall_cool_off, e$
          coolant_on = zero
          ]
        else,
          [
          local_int = zero
          coolantx = zero
          while local_int < 20 & coolant_on > 0,
            [
            coolantx = and(2^local_int, coolant_on)
            local_int = local_int + one
            if coolantx > zero,
              [
              coolantx = local_int
              pbld, n$, scoolantx, e$
              ]
            coolantx = zero
            ]
          coolant_on = zero
          ]
        ]
      #cc_pos is reset in the toolchange here
      cc_pos$ = zero
      gcode$ = zero
      pbld, n$, sccomp, *sm05, psub_end_mny, e$
      pbld, n$, sgabsinc, sgcode, *sg28ref, scoolant, e$ #"Z0.",
      if nextop$ = 1003 | tlchg_home, pbld, n$, e$ #*sg28ref, "X0.", "Y0.", e$
      absinc$ = sav_absinc
      coolant$ = sav_coolant

peof0$           #End of file for tool zero
      peof$

peof$            #End of file for non-zero tool
      pretract
      comment$
      if stagetool = 1 & stagetltype = 2, pbld, n$, *first_tool$, e$
      n$, "M30", e$
      mergesub$
      clearsub$
      mergeaux$
      clearaux$
      "%", e$

pwcs            # Configuração do sistemas de eixos do tipo G15
      if mi1$ = one,
        [
        sav_frc_wcs = force_wcs
        if sub_level$ > 0, force_wcs = zero
        if workofs$ <> prv_workofs$ | (force_wcs & toolchng),
          [
               if ((workofs$>0&workofs$<51) | (workofs$>100&workofs$<151)),
               [
            h_wcs = workofs$
            *sg15, *h_wcs
               ]
          else,
          [
          "(SISTEMA DE EIXOS G15 DEVE SER : H01-H50 ou H101-H150 ..... POST TERMINADO)",
e$
          exitpost$
          ]
          ]
        force_wcs = sav_frc_wcs
        !workofs$
        ]

# ANTIGO SISTEMA G54
#if mi1$ > one,
        #[
        #sav_frc_wcs = force_wcs
        #if sub_level$ > 0, force_wcs = zero
        #if workofs$ <> prv_workofs$ | (force_wcs & toolchng),
          #[
          #if workofs$ < 6,
```

```
            #[
            #g_wcs = workofs$ + 54
            #*g_wcs
            #]
          #else,
            #[
            #p_wcs = workofs$ - five
            #"G54.1", *p_wcs
            #]
          #]
        #force_wcs = sav_frc_wcs
        #!workofs$
        #]
pgear           #Find spindle gear from lookup table
      if use_gear = one,
        [
        gear = frange (one, speed)
        *gear
        ]

#Toolchange setup
pspindchng      #Spindle speed change
      if prv_spdir2 <> spdir2 & prv_speed <> zero, pbld, n$, *sm05, e$
      if prv_speed <> speed | prv_spdir2 <> spdir2,
        [
        if speed, pbld, n$, *speed, *spindle, pgear, e$
        ]
      !speed, !spdir2

pspindle        #Spindle speed calculations for RPM
      speed = abs(ss$)
      if speed,
        [
        if speed > maxss$, speed = maxss$
        if speed < min_speed, speed = min_speed
        ]
      spdir2 = fsg3(spdir$)

pq$              #Setup post based on switch settings
      stagetool = bldnxtool$  #Set stagetool from CD setting

ptoolend$        #End of tool path, before reading new tool data
      !speed, !spdir2

ptlchg1002$      #Call at actual toolchange, end last path here
      if op_id$ <> prv_op_id$, pset_mach   #Read machine def parameters
      pspindle
      whatline$ = four #Required for vector toolpaths
      if gcode$ = 1000,
        [
        #Null toolchange
        ]
      else,
        [
        #Toolchange and Start of file
        if gcode$ = 1002,
          [
          #Actual toolchange
          pretract
          ]
        if stagetool = one, prv_next_tool$ = m_one
        prv_xia = vequ(xh$)
        prv_feed = c9k
        ]
      !op_id$

# ------------------------------------------------------------------------
# Motion NC output
# ------------------------------------------------------------------------
#The variables for absolute output are xabs, yabs, zabs.
#The variables for incremental output are xinc, yinc, zinc.
# ------------------------------------------------------------------------
prapidout       #Output to NC of linear movement - rapid
      pcan1, pbld, n$, sgplane, `sgcode, sgabsinc, pccdia,
        pxout, pyout, pzout, strcantext, scoolant, e$
      #Modify following line to customize output for high-speed toolpath
```

```
      #tool inspection/change points
      if rpd_typ$ = 7, pbld, n$, "M00", "(TOOL INSPECTION POINT - POST CUSTOMIZATION
REQUIRED)", e$

plinout         #Output to NC of linear movement - feed
      pcan1, pbld, n$, sgplane, `sgcode, sgabsinc, pccdia,
        pxout, pyout, pzout, feed, strcantext, scoolant, e$
      #Modify following line to customize output for high-speed toolpath
      #tool inspection/change points
      if rpd_typ$ = 7, pbld, n$, "M00", "(TOOL INSPECTION POINT - POST CUSTOMIZATION
REQUIRED)", e$

pcirout         #Output to NC of circular interpolation
      pcan1, pbld, n$, sgplane, sgcode, sgabsinc, pccdia,
        pxout, pyout, pzout, parc, feed, strcantext, scoolant, e$

pcom_moveb      #Common motion preparation routines, before
      pxyzcout
      ps_inc_calc

pncoutput       #Movement output
      pcom_moveb
      comment$
      pcan
      if gcode$ = zero, prapidout
      if gcode$ = one, plinout
      if gcode$ > one & gcode$ < four, pcirout
      if mr_rt_actv, #Restore absolute/incremental for G51/G68
        [
        absinc$ = sav_absinc
        mr_rt_actv = zero
        ]
      pcom_movea

pcom_movea      #Common motion preparation routines, after
      pcan2
      pe_inc_calc

pdwl_spd$        #Call from NCI gcode 4
      pspindle
      comment$
      pspindchng
      pcan
      if fmtrnd(dwell$), pcan1, pbld, n$, *sgcode, *dwell$, strcantext, e$
      else, pcan1, pbld, n$, strcantext, e$
      pcan2

prapid$          #Output to NC of linear movement - rapid
      pncoutput

pzrapid$         #Output to NC of linear movement - rapid Z only
      pncoutput

plin$            #Output to NC of linear movement - feed
      pncoutput

pz$              #Output to NC of linear movement - feed Z only
      pncoutput

pmx$             #Output to NC of vector NCI
      pncoutput

pcir$            #Output to NC of circular interpolation
      pncoutput

# ------------------------------------------------------------------------
# Motion output components
# ------------------------------------------------------------------------
pbld            #Canned text - block delete
      if bld, '/'

pfbld           #Force - block delete
      "/"

pccdia          #Cutter Compensation
      #Force Dxx#
```

```
      if prv_cc_pos$ <> cc_pos$ & cc_pos$, prv_tloffno$ = c9k
      sccomp
      if cc_pos$, tloffno$

pfxout          #Force X axis output
      if (xabs>x_limits_max | xabs<x_limits_min), errox = 1
      else, errox = 0
      perrox
      if absinc$ = zero,
      [
      xabs, *serrox, !xinc
      perroexit
      ]
      else, *xinc, !xabs

pxout           #X output

      if (xabs>x_limits_max | xabs<x_limits_min), errox = 1
      else, errox = 0
      perrox
      if absinc$ = zero,
      [
      xabs, serrox, !xinc
      perroexit
      ]
      else, xinc, !xabs

pfyout          #Force Y axis output

      if (yabs>y_limits_max | yabs<y_limits_min), erroy = 1
      else, erroy = 0
      perroy
      if absinc$ = zero,
      [
      yabs, *serroy, !yinc
      perroexit
      ]
      else, *yinc, !yabs

pyout           #Y output

      if (yabs>y_limits_max | yabs<y_limits_min), erroy = 1
      else, erroy = 0
      perroy
      if absinc$ = zero,
      [
      yabs, serroy, !yinc
      perroexit
      ]
      else, yinc, !yabs

pfzout          #Force Z axis output

      if                ((zabs-tool_ol-tool_hl-t_orgin_z$)>z_limits_max                |
(zabs+tool_ol+tool_hl+t_orgin_z$)<z_limits_min), erroz = 1
      else, erroz = 0
      perroz
      if absinc$ = zero,
      [
      *zabs, *serroz, !zinc
      perroexit
      ]
      else, *zinc, !zabs

pzout           #Z output

      if                ((zabs-tool_ol-tool_hl-t_orgin_z$)>z_limits_max                |
(zabs+tool_ol+tool_hl+t_orgin_z$)<z_limits_min), erroz = 1
      else, erroz = 0
      perroz
      if absinc$ = zero,
      [
      zabs, serroz, !zinc
      perroexit
      ]
      else, zinc, !zabs
```

```
parc            #Select the arc output

      if (plane$ = zero & (arctype$ = one | arctype$ = four)) |   #XY Plane
        (plane$ = one & (arctypeyz$ = one | arctypeyz$ = four)) | #YZ Plane
        (plane$ = two & (arctypexz$ = one | arctypexz$ = four)),  #XZ Plane
        [
        result = newfs(two, iout)
        result = newfs(two, jout)
        result = newfs(two, kout)
        ]
      else,
        [
        result = newfs(three, iout)
        result = newfs(three, jout)
        result = newfs(three, kout)
        ]
      if (plane$ = 0 & arctype$ < five) | (plane$ = 1 & arctypeyz$ < five) |
        (plane$ = 2 & arctypexz$ < five) | full_arc_flg$ | arc_pitch$,
        [
        #Arc output for IJK
        # If you do NOT want to force out the I,J,K values,
        # remove the "*" asterisks on the *i, *j, *k 's below...
        if plane$ = zero, *iout, *jout, kout #XY plane code - G17
        if plane$ = one, iout, *jout, *kout  #YZ plane code - G19
        if plane$ = two, *iout, jout, *kout  #XZ plane code - G18
        !i$, !j$, !k$
        ]
      else,
        [
        #Arc output for R
        if abs(sweep$)<=180 | (plane$ = 0 & arctype$ = five) | (plane$ = 1 & arctypeyz$
= five) |
          (plane$ = 2 & arctypexz$ = five), result = nwadrs(srad, arcrad$)
        else, result = nwadrs(srminus, arcrad$)
        *arcrad$
        ]

perrox
      if errox = 0, serrox = sblank
      else,
        [
        result = newfs(two, xabs)
        xabs
        if errox = 1 & xabs<>xabs_b,
          [
          serrox = "(<XFDL)"
          xabs_b = xabs
          ]
        else, serrox = sblank
        ]

perroy
      if erroy = 0, serroy = sblank
      else,
        [
        result = newfs(two, yabs)
        yabs
        if erroy = 1 & yabs<>yabs_b,
          [
          serroy = "(<YFDL)"
          yabs_b = yabs
          ]
        else, serroy = sblank
        ]

perroz
      if erroz = 0, serroz = sblank
      else,
        [
        result = newfs(two, zabs)
        zabs
        if erroz = 1 & zabs<>zabs_b,
          [
          serroz = "(<ZFDL)"
          erroz = 0
```

```
          zabs_b = zabs
          ]
        else, serroz = sblank
        ]


perroexit
      if mi4$ = 0, sblank
      if (mi4$ = 1 & (errox = 1 | erroy = 1 | erroz = 1)), serroexit, e$, exitpost$
      else, sblank

# ------------------------------------------------------------------------
# Drilling
# ------------------------------------------------------------------------
pdrlcommonb     #Canned Drill Cycle common call, before
      if initht$ <> refht$, drillref = zero
      else, drillref = one
      if gcode$ = 81,
        [
        result = newfs(two, zinc)
        if drillcyc$ = three | drillcyc$ = 7, drlgsel = fsg1(-ss$) + drillcyc$ * two
        else, drlgsel = fsg2(dwell$) + drillcyc$ * two
        prv_refht_a = c9k
        prv_refht_i = c9k
        prv_dwell$ = zero
        prv_shftdrl$ = zero
        ]
      z$ = depth$
      prv_zia = refht$
      pcom_moveb
      feed = fr_pos$
      comment$
      pcan

prdrlout        #R drill position
      refht_a = refht$
      refht_i = refht$ - initht$
      if absinc$ = zero, refht_a, !refht_i
      else, refht_i, !refht_a

pdrill$          #Canned Drill Cycle
      pdrlcommonb
      pcan1, pbld, n$, *sgdrlref, *sgdrill, pxout, pyout, pfzout,
        prdrlout, dwell$, *feed, strcantext, e$
      pcom_movea

ppeck$           #Canned Peck Drill Cycle
      pdrlcommonb
      pcan1, pbld, n$, *sgdrlref, *sgdrill, pxout, pyout, pfzout,
        prdrlout, *peck1$, *feed, strcantext, e$
      pcom_movea

pchpbrk$         #Canned Chip Break Cycle
      pdrlcommonb
      pcan1, pbld, n$, *sgdrlref, *sgdrill, pxout, pyout, pfzout,
        prdrlout, *peck1$, *feed, strcantext, e$
      pcom_movea

ptap$            #Canned Tap Cycle
      pdrlcommonb
      result = newfs(17, feed)  # Set for tapping Feedrate format
      pcan1, pbld, n$, *sgdrlref, *sgdrill, pxout, pyout, pfzout,
        prdrlout, *feed, strcantext, e$
      pcom_movea

pbore1$          #Canned Bore #1 Cycle
      pdrlcommonb
      pcan1, pbld, n$, *sgdrlref, *sgdrill, pxout, pyout, pfzout,
        prdrlout, dwell$, *feed, strcantext, e$
      pcom_movea

pbore2$          #Canned Bore #2 Cycle
      pdrlcommonb
      pcan1, pbld, n$, *sgdrlref, *sgdrill, pxout, pyout, pfzout,
        prdrlout, *feed, strcantext, e$
      pcom_movea
```

```
pmisc1$          #Canned Fine Bore (shift) Cycle
      pdrlcommonb
      pcan1, pbld, n$, *sgdrlref, *sgdrill, pxout, pyout, pfzout,
        prdrlout, shftdrl$, dwell$, *feed, strcantext, e$
      pcom_movea

pmisc2$          #Canned Rigid Tapping Cycle
      pdrlcommonb
      #RH/LH based on spindle direction
      pbld, n$, sg95, e$
      pbld, n$, sm29, *speed, e$
      if met_tool$, pitch = n_tap_thds$  #Tap pitch (mm  per thread)
      else, pitch = 1/n_tap_thds$        #Tap pitch (inches per thread)
      pcan1, pbld, n$, *sgdrlref, *sgdrill, pxout, pyout, pfzout,
        prdrlout, *pitch, !feed, strcantext, e$
      pcom_movea

pdrlcst$         #Custom drill cycles 8 - 19 (user option)
      #Use this postblock to customize drilling cycles 8 - 19
      if drillcyc$ = 8, pdrlcst8
      else,
        [
        pdrlcommonb
        sopen_prn, "CUSTOMIZABLE DRILL CYCLE - NOT CONFIGURED - FIRST HOLE", sclose_prn,
e$
        pcom_movea
        ]

pdrlcst8         #Custom drill cycle 8 - example custom cycle
      pdrlcommonb
      sopen_prn, "CUSTOMIZABLE DRILL CYCLE EXAMPLE - FIRST HOLE", sclose_prn, e$
      pcan1, pbld, n$, *sgdrlref, *sgdrill, pxout, pyout, pfzout,
        prdrlout, shftdrl$, dwell$, *feed, strcantext, e$
      pcom_movea

# Additional Holes
pdrill_2$        #Canned Drill Cycle, additional points
      pdrlcommonb
      pcan1, pbld, n$, pxout, pyout, pzout, prdrlout, feed, strcantext, e$
      pcom_movea

ppeck_2$         #Canned Peck Drill Cycle
      pdrill_2$

pchpbrk_2$       #Canned Chip Break Cycle
      pdrill_2$

ptap_2$          #Canned Tap Cycle
      pdrill_2$

pbore1_2$        #Canned Bore #1 Cycle
      pdrill_2$

pbore2_2$        #Canned Bore #2 Cycle
      pdrill_2$

pmisc1_2$        #Canned Fine Bore (shift) Cycle
      pdrill_2$

pmisc2_2$        #Canned Rigid Tapping Cycle
      pdrlcommonb
      pcan1, pbld, n$, pxout, pyout, pzout, prdrlout, strcantext, e$
      pcom_movea

pdrlcst_2$       #Custom drill cycles 8 - 19, additional points (user option)
      #Use this postblock to customize drilling cycles 8 - 19
      if drillcyc$ = 8, pdrlcst8_2
      else,
        [
        sopen_prn, "CUSTOMIZABLE DRILL CYCLE - NOT CONFIGURED - NEXT HOLE", sclose_prn,
e$
        pdrill_2$
        ]

pdrlcst8_2       #Custom drill cycle 8 - example custom cycle
```

```
      sopen_prn, "CUSTOMIZABLE DRILL CYCLE EXAMPLE - NEXT HOLE", sclose_prn, e$
      pdrill_2$

pcanceldc$       #Cancel canned drill cycle
      result = newfs(three, zinc)
      z$ = initht$
      prv_zia = initht$
      pxyzcout
      !zabs, !zinc
      prv_gcode$ = zero
      pcan
      pcan1, pbld, n$, sg80, strcantext, e$
      if drillcyc$ = 7, pbld, n$, sg94, e$
      pcan2

# ------------------------------------------------------------------------
#Subprogram postblocks
#sub_trnstyp - 0=mirror, 1=rotate, 2=scale, 3=translate
#sub_trnmthd (mirror) - 0=X axis, 1=Y axis, 2=line
#sub_trnmthd (rotate) - 0=tplane, 1=tplane origin only, 2=coordinates
# ------------------------------------------------------------------------
psub_call_m$     #Call to main level, single tool
      psub_call_trans

psub_call_mm$    #Call to main level, multiple tools
      psub_call_trans

psub_call_trans #Translate level calls from toolchange, user
      #if mi1$ = two, result = mprint(shomeserror)
      sav_absinc = absinc$
      #Mirror or Rotate Coord's
      if sub_trnstyp$ = zero | (sub_trnstyp$ = one & mr_rt_actv),
        [
        #The original pattern is not mirrored or rotated
        if sub_sec_no$,
          [
          absinc$ = zero
          if sub_trnstyp$, psub_rotate
          else, psub_mirror
          ]
        mr_rt_actv = three
        ]
      else, #Translate
        [
        if sub_mny_t$,
          [
          if mi1$ = one, absinc$ = zero
          pbld, n$, *sgcode, *sgabsinc, pwcs, pfxout, pfyout, pfzout, e$
          pe_inc_calc
          ps_inc_calc
          ]
        ]
      absinc$ = sav_absinc
      #result = nwadrs(strosub, main_prg_no$)
      if progno$ = main_prg_no$, result = mprint(sprgnerror)
      osub_prog_n = main_prg_no$
      result = newfs(osub_prog_n, 4)
      pbld, n$, "CALL ", *osub_prog_n, e$
      prv_feed = c9k #Force feed in sub

psub_mirror     #Mirror start code, user
      #Mirror Y axis
      if sub_trnmthd$, pbld, n$, *sgabsinc, strns_mir_on, *sub_trnsx$, e$
      #Mirror X axis
      else, pbld, n$, *sgabsinc, strns_mir_on, *sub_trnsy$, e$

psub_rotate     #Rotate start code, user
      pbld, n$, *sgcode, *sgabsinc, strns_rot_on, *sub_trnsx$, *sub_trnsy$,
        [absinc$ = one], *sgabsinc, e$

psub_st_m$       #Header in main level
      #result = nwadrs(stro, main_prg_no$)
      " ", e$
      *osub_prog_n, e$
      #G51/G68 requires absolute position on first move
      if mr_rt_actv & absinc$ = one,
```

```
        [
        sav_absinc = absinc$
        absinc$ = zero
        prv_absinc$ = m_one
        prv_xabs = m_one
        prv_yabs = m_one
        ]
      else, pbld, n$, sgabsinc, e$

psub_end_m$      #End in main level
      n$, "RTS", e$
      prv_absinc$ = m_one
      #Reset update variables for subs at main level
      #Mirror or Rotate cancel, output is forced
      if (sub_trnstyp$ = zero & esub_sec_no$ > zero)
        | (sub_trnstyp$ = one & esub_sec_no$ = esub_totl_no$-one
        & sub_trnmthd$ = two),
        [
        subout$ = zero
        no_nc_out$ = m_one
        sav_absinc = absinc$
        #Mirror cancel
        if sub_trnstyp$ = zero,
          [
          absinc$ = zero
          pbld, n$, *sgabsinc, strns_mir_off, *sub_trnsx$, *sub_trnsy$, e$
          ]
        else, #Rotate cancel
          [
          pbld, n$, strns_rot_off, e$
          ]
        absinc$ = sav_absinc
        no_nc_out$ = zero
        ]
      end_sub_mny = sub_mny_t$

psub_end_mny    #End in main level for many tools sub, user
      #Check for coming out of xform with stage tool.
      if end_sub_mny & stagetool = one,
        [
        *t$
        end_sub_mny = zero
        ]

psub_call_s$     #Call to sub level
      #result = nwadrs(strosub, sub_prg_no$)
      sub_prg_no$ = sub_prg_no$ + 1000 #Add sub number offset
      if progno$ = sub_prg_no$, result = mprint(sprgnerror)
      osub_prog_n = main_prg_no$
      result = newfs(osub_prog_n, 4)
      pbld, n$, "CALL ", *osub_prog_n, e$

psub_st_s$       #Header in sub leveln
      #result = nwadrs(stro, sub_prg_no$)
      " ", e$
      *osub_prog_n, e$
      pbld, n$, sgabsinc, e$

psub_end_s$      #End in sub level
      n$, "RTS", e$
      prv_absinc$ = -1

# ------------------------------------------------------------------------
# Canned Text
# ------------------------------------------------------------------------
pcan            #Canned text - before output call
      strcantext = sblank
      if cant_no$ > zero,
        [
        if cant_pos1$ = zero | cant_pos1$ = three, pcant_1
        if cant_pos2$ = zero | cant_pos2$ = three, pcant_2
        if cant_pos3$ = zero | cant_pos3$ = three, pcant_3
        if cant_pos4$ = zero | cant_pos4$ = three, pcant_4
        if cant_pos5$ = zero | cant_pos5$ = three, pcant_5
        if cant_pos6$ = zero | cant_pos6$ = three, pcant_6
        if cant_pos7$ = zero | cant_pos7$ = three, pcant_7
```

```
        if cant_pos8$ = zero | cant_pos8$ = three, pcant_8
        if cant_pos9$ = zero | cant_pos9$ = three, pcant_9
        if cant_pos10$ = zero | cant_pos10$ = three, pcant_10
        if cant_pos11$ = zero | cant_pos11$ = three, pcant_11
        if cant_pos12$ = zero | cant_pos12$ = three, pcant_12
        if cant_pos13$ = zero | cant_pos13$ = three, pcant_13
        if cant_pos14$ = zero | cant_pos14$ = three, pcant_14
        if cant_pos15$ = zero | cant_pos15$ = three, pcant_15
        if cant_pos16$ = zero | cant_pos16$ = three, pcant_16
        if cant_pos17$ = zero | cant_pos17$ = three, pcant_17
        if cant_pos18$ = zero | cant_pos18$ = three, pcant_18
        if cant_pos19$ = zero | cant_pos19$ = three, pcant_19
        if cant_pos20$ = zero | cant_pos20$ = three, pcant_20
        pbld, n$, strcantext, e$
        strcantext = sblank
        ]

pcan1           #Canned text - with move
      strcantext = sblank
      if cant_no$ > zero,
        [
        if cant_pos1$ = one | cant_pos1$ = four, pcant_1
        if cant_pos2$ = one | cant_pos2$ = four, pcant_2
        if cant_pos3$ = one | cant_pos3$ = four, pcant_3
        if cant_pos4$ = one | cant_pos4$ = four, pcant_4
        if cant_pos5$ = one | cant_pos5$ = four, pcant_5
        if cant_pos6$ = one | cant_pos6$ = four, pcant_6
        if cant_pos7$ = one | cant_pos7$ = four, pcant_7
        if cant_pos8$ = one | cant_pos8$ = four, pcant_8
        if cant_pos9$ = one | cant_pos9$ = four, pcant_9
        if cant_pos10$ = one | cant_pos10$ = four, pcant_10
        if cant_pos11$ = one | cant_pos11$ = four, pcant_11
        if cant_pos12$ = one | cant_pos12$ = four, pcant_12
        if cant_pos13$ = one | cant_pos13$ = four, pcant_13
        if cant_pos14$ = one | cant_pos14$ = four, pcant_14
        if cant_pos15$ = one | cant_pos15$ = four, pcant_15
        if cant_pos16$ = one | cant_pos16$ = four, pcant_16
        if cant_pos17$ = one | cant_pos17$ = four, pcant_17
        if cant_pos18$ = one | cant_pos18$ = four, pcant_18
        if cant_pos19$ = one | cant_pos19$ = four, pcant_19
        if cant_pos20$ = one | cant_pos20$ = four, pcant_20
        ]
      if cstop$, strcantext = strcantext + sm00
      if cgstop$, strcantext = strcantext + sm01
      #Output of strcantext occurs at the end of the output line

pcan2           #Canned text - after output call
      strcantext = sblank
      if cant_no$ > zero,
        [
        if cant_pos1$ = two | cant_pos1$ = five, pcant_1
        if cant_pos2$ = two | cant_pos2$ = five, pcant_2
        if cant_pos3$ = two | cant_pos3$ = five, pcant_3
        if cant_pos4$ = two | cant_pos4$ = five, pcant_4
        if cant_pos5$ = two | cant_pos5$ = five, pcant_5
        if cant_pos6$ = two | cant_pos6$ = five, pcant_6
        if cant_pos7$ = two | cant_pos7$ = five, pcant_7
        if cant_pos8$ = two | cant_pos8$ = five, pcant_8
        if cant_pos9$ = two | cant_pos9$ = five, pcant_9
        if cant_pos10$ = two | cant_pos10$ = five, pcant_10
        if cant_pos11$ = two | cant_pos11$ = five, pcant_11
        if cant_pos12$ = two | cant_pos12$ = five, pcant_12
        if cant_pos13$ = two | cant_pos13$ = five, pcant_13
        if cant_pos14$ = two | cant_pos14$ = five, pcant_14
        if cant_pos15$ = two | cant_pos15$ = five, pcant_15
        if cant_pos16$ = two | cant_pos16$ = five, pcant_16
        if cant_pos17$ = two | cant_pos17$ = five, pcant_17
        if cant_pos18$ = two | cant_pos18$ = five, pcant_18
        if cant_pos19$ = two | cant_pos19$ = five, pcant_19
        if cant_pos20$ = two | cant_pos20$ = five, pcant_20
        pbld, n$, strcantext, e$
        strcantext = sblank
        ]

pcant_1         #Canned text - output call
      cant_pos = cant_pos1$
```

```
      cantext$ = cant_val1$
      pcant_out

pcant_2         #Canned text - output call
      cant_pos = cant_pos2$
      cantext$ = cant_val2$
      pcant_out

pcant_3         #Canned text - output call
      cant_pos = cant_pos3$
      cantext$ = cant_val3$
      pcant_out

pcant_4         #Canned text - output call
      cant_pos = cant_pos4$
      cantext$ = cant_val4$
      pcant_out

pcant_5         #Canned text - output call
      cant_pos = cant_pos5$
      cantext$ = cant_val5$
      pcant_out

pcant_6         #Canned text - output call
      cant_pos = cant_pos6$
      cantext$ = cant_val6$
      pcant_out

pcant_7         #Canned text - output call
      cant_pos = cant_pos7$
      cantext$ = cant_val7$
      pcant_out

pcant_8         #Canned text - output call
      cant_pos = cant_pos8$
      cantext$ = cant_val8$
      pcant_out

pcant_9         #Canned text - output call
      cant_pos = cant_pos9$
      cantext$ = cant_val9$
      pcant_out

pcant_10        #Canned text - output call
      cant_pos = cant_pos10$
      cantext$ = cant_val10$
      pcant_out

pcant_11        #Canned text - output call
      cant_pos = cant_pos11$
      cantext$ = cant_val11$
      pcant_out

pcant_12        #Canned text - output call
      cant_pos = cant_pos12$
      cantext$ = cant_val12$
      pcant_out

pcant_13        #Canned text - output call
      cant_pos = cant_pos13$
      cantext$ = cant_val13$
      pcant_out

pcant_14        #Canned text - output call
      cant_pos = cant_pos14$
      cantext$ = cant_val14$
      pcant_out

pcant_15        #Canned text - output call
      cant_pos = cant_pos15$
      cantext$ = cant_val15$
      pcant_out

pcant_16        #Canned text - output call
      cant_pos = cant_pos16$
      cantext$ = cant_val16$
```

```
      pcant_out

pcant_17        #Canned text - output call
      cant_pos = cant_pos17$
      cantext$ = cant_val17$
      pcant_out

pcant_18        #Canned text - output call
      cant_pos = cant_pos18$
      cantext$ = cant_val18$
      pcant_out

pcant_19        #Canned text - output call
      cant_pos = cant_pos19$
      cantext$ = cant_val19$
      pcant_out

pcant_20        #Canned text - output call
      cant_pos = cant_pos20$
      cantext$ = cant_val20$
      pcant_out

pcant_out       #Canned text - build the string for output
      #Assign string select type outputs
      if cant_pos < three, #cant_pos indicates canned text output
        [
        if cantext$ = three, bld = one
        if cantext$ = four, bld = zero
        #Build the cantext string
        if cantext$ = one, strcantext = strcantext + sm00
        if cantext$ = two, strcantext = strcantext + sm01
        if cantext$ > four,
          [
          strtextno = no2str(cantext$)
          strcantext = strcantext + strm + strtextno
          ]
        ]
      else, #cant_pos indicates coolant output
        [
        coolant_bin = flook (two, cantext$) #Create binary value for each coolant using
lookup table
        if frac(cantext$/two),  # coolant off
          [
          if all_cool_off,
            [
            if coolant_on, pbld, n$, sall_cool_off, e$
            coolant_on = zero
            ]
          else,
            [
            if coolant_on > 0,
              [
              coolant_on  =  coolant_on  -  coolant_bin/2  #Odd  =  off  command,  subtract
appropriate binary value.
              coolantx = cantext$ - 50                    #Create a coolantx value for
string select
              pbld, n$, *scoolantx, e$
              ]
            ]
          ]
        else,                                          #Even = on command
          [   #Determine if this coolant is already on
          local_int = zero
          coolantx = zero
          suppress = zero
          while local_int < 20 & coolant_on > 0,
            [
            result2 = and(2^local_int, coolant_on)
            local_int = local_int + one
            if result2 = coolant_bin, suppress = one
            ]
          if suppress <> 1, #Don't output an on code for a coolant that is already on
            [
            coolant_on = coolant_on + coolant_bin #Maintain binary sum of all coolants
currently on
```

```
            coolantx = cantext$ - 50                #Create a coolantx value for string
select
            pbld, n$, *scoolantx, e$
            ]
          ]
        ]

# ------------------------------------------------------------------------
# Position calculations, generally these do not need to be modified
# ------------------------------------------------------------------------
pmiscint$        #Capture the top level absinc for subprograms
      if sub_level$ <= zero, absinc$ = mi2$

pcheckaxis      #Check for valid rotary axis
      #If selected axis combination has more than 0 rotary axes and toolpath has
rotation
      if (mill5$ | rotary_type$ = 1 | rotary_type$ = 2) & rotaxerror = 1, [if
mprint(srotaxerror, 2) = 2, exitpost$]

pxyzcout        #Map coordinates
      xabs = vequ(x$)    #Retrieve X,Y,Z positions into xabs,yabs,zabs
      iout = vequ(i$)
      feed = fr_pos$
      if feed > maxfeedpm, feed = maxfeedpm

#Incremental calculations
ps_inc_calc     #Incremental calculations, start
      xia = fmtrnd(xabs)
      yia = fmtrnd(yabs)
      zia = fmtrnd(zabs)
      xinc = vsub (xia, prv_xia)

pe_inc_calc     #Incremental calculations, end
      !xia, !yia, !zia
      !x$, !y$, !z$, !cc_pos$

# ------------------------------------------------------------------------
# Parameter read postblocks:
# ------------------------------------------------------------------------
pprep$             #Pre-process postblock - Allows post instructions after the post is
parsed but before the NC and NCI file are opened.
#DO NOT ATTEMPT TO OUTPUT TO THE NC FILE IN THIS POSTBLOCK (OR ANY POSTBLOCKS YOU MAY
CALL FROM HERE) BECAUSE THE NC OUTPUT FILE IS NOT YET OPENED!
      rd_cd$       #Read CD Parameters
      rd_mch_ent_no$ = 0  #Read only the machine base parameters (use to collect common
parameters from CNC_MACHINE_TYPE)
      rd_md$       #Read machine definition parameters
      rd_mch_ent_no$ = -1
      rd_md$


pwrtt$          #Pre-read NCI file
      if tool_info > 1 & t$ > 0 & gcode$ <> 1003, ptooltable
      if ra_type$, ra_error = 1  #Aggregates NOT supported in this post




pwrttparam$     #Pre-read parameter data
      #"pwrttparam", ~prmcode$, ~sparameter$, e$
      if prmcode$ = 15346, comp_type = rpar(sparameter$, 1) #Cutter compensation type -
0=computer, 1=control, 2=wear, 3=reverse wear, 4=off
      if prmcode$ = 10010, xy_stock = rpar(sparameter$, 1)  #Capture stock to leave (XY)
      if prmcode$ = 10068, z_stock = rpar(sparameter$, 1)   #Capture stock to leave (Z)




pparameter$     #Read operation parameters
      #rd_params is used to call pparameter postblock and read the parameters of the
operation specified in rd_param_op_no
      #"pparameter", ~prmcode$, ~sparameter$, e$
      if prmcode$ = 20600, comp_n = rpar(sparameter$, 1)
      if (comp_n>0), [comp_f=comp_n]
```

```
      if (comp_i=0 & comp_n>0), [comp_i = comp_n, ent_i = comp_i]

      if prmcode$ = 20007, tool_ol = rparsngl(sparameter$, 3)
      if prmcode$ = 20007, tool_hl = rparsngl(sparameter$, 7)
      if (tool_ol > 0 & fto = 0),
      [
       fto=tool_ol+tool_hl
      ]
      if (tool_d = 0),
      [
       if prmcode$ = 20004, tool_d = rparsngl(sparameter$, 5)
      ]
      #*tool_ol, e$
      #*tool_hl, e$

# ------------------------------------------------------------------------
# Parameter lookup tables - You must adjust the size value if you add or remove any
parameters to/from these tables!
# ------------------------------------------------------------------------
# Machine Definition Parameters
fprmtbl 17000   4    #Table Number, Size
#       Param   Variable to load value into
        17605   min_speed    #Minimum spindle speed
        17055   maxfeedpm    #Limit for feed in inch/min
        17063   maxfeedpm_m  #Limit for feed in mm/min
        17101   all_cool_off #First coolant off command shuts off ALL coolant options

# Control Definition Parameters
fprmtbl 18000   0    #Table Number, Size
#       Param   Variable to load value into

# Toolpath Group Parameters
fprmtbl 19000   0    #Table Number, Size
#       Param   Variable to load value into

# ------------------------------------------------------------------------
pset_mach       #Set post switches by reading machine def parameters
      rot_ax_cnt = 0
      rotaxerror = 0
      #maxfeedpm = 999999        #Uncomment these variables to force use of machine def
values as initial lowest max feedrate value
      #maxfeedpm_m = 9999999    #Otherwise the default (post) initialization setting is
used as initial value
      !maxfeedpm, !maxfeedpm_m

      rd_mch_ent_no$ = syncaxis$   #Retrieve machine parameters based on current axis
combination - read from .nci G950 line
      rd_md$       #Read machine definition parameters - calls pmachineinfo$

      #We only need this set at toolchange (and start of file).  No need to set them
each time a user may call rd_md
      if met_tool$ = 1, maxfeedpm = maxfeedpm_m         #Set limit for feed in mm/min

# ------------------------------------------------------------------------
# Machine definition and control definition parameter capture:
# ------------------------------------------------------------------------
pmachineinfo$   #Machine information parameters postblock
      #rd_md is used to call pmachineinfo postblock and read the parameters of the
selected axis
      #combination machine entity set in rd_mch_ent_no.
      #rd_cd is used to call pmachineinfo postblock and read the active control
definition parameters
      #rd_tlpathgrp is used to call pmachineinfo postblock and read the active toolpath
group parameters
      #"-->pmachineinfo", ~prmcode$, "  ", ~sparameter$, e$  #Do not uncomment if being
called from pprep$ - see pprep comment

      if codestart=1,
         [
      if prmcode$ = 17390, saxis_lbl = sparameter$
      if x_set = 0,
      [
      if (saxis_lbl = strx),
      [
       if prmcode$ = 17726, [X_limits_max = rpar(sparameter$, 1), x_set = 1]
       if prmcode$ = 17725, X_limits_min = rpar(sparameter$, 1)
```

```
      ]
      ]
      if y_set = 0,
      [
      if (saxis_lbl = stry),
      [
       if prmcode$ = 17726, [Y_limits_max = rpar(sparameter$, 1), y_set = 1]
       if prmcode$ = 17725, Y_limits_min = rpar(sparameter$, 1)
      ]
      ]
      if z_set = 0,
      [
      if (saxis_lbl = strz),
      [
       if prmcode$ = 17726, [Z_limits_max = rpar(sparameter$, 1), z_set = 1]
       if prmcode$ = 17725, Z_limits_min = rpar(sparameter$, 1)
      ]
      ]
      if (x_set=0|y_set=0|z_set=0),
      [ent_i=ent_i+1, plimits]
      else,
      ]

      #Read parameter lookup tables -
      if prmcode$ >= 17000 & prmcode$ < 18000, result = fprm(17000) #Run the parameter
table for Machine Definition Parameters
      #Leave lines below commented until you enter values in related lookup tables
      #if prmcode$ >= 18000 & prmcode$ < 19000, result = fprm(18000) #Run the parameter
table for Control Definition Parameters
      #if prmcode$ >= 19000 & prmcode$ < 19900, result = fprm(19000) #Run the parameter
table for Toolpath Group Parameters

      #Count rotary axis and output error message if one is found in the active axis
combination
      if prmcode$ = 19958,
        [
        component_type = rpar(sparameter$, 1)  #Component type
        if component_type = 5,
          [
          rot_ax_cnt = rot_ax_cnt + 1  #Rotary component
          if rot_ax_cnt = 1, rotaxerror = rotaxerror + 1    #Post only supports 1 rotary
per axis combination
          ]
        ]

      #Read Linear Axis parameters - capture lowest feedrate value of all linear axis
      if maxfeedpm > prv_maxfeedpm, maxfeedpm = prv_maxfeedpm
      if maxfeedpm_m > prv_maxfeedpm_m, maxfeedpm_m = prv_maxfeedpm_m
      !maxfeedpm, !maxfeedpm_m

# ------------------------------------------------------------------------
# POST TEXT
# ------------------------------------------------------------------------
[CTRL_MILL|OKUMA MC-40VA]
[misc integers]
1. "Tipo de sistemas de coordenadas [1=G15, 2=G92]"//1
2. "Programa em modo ABSoluto ou INCremental [0=ABS, 1=INC]"
3. "Tipo de referência de retorno [0=G28, 1=G30]"//1
4. "Pós-processador é interrompido em caso de erro [0=NO, 1=YES] "//1
5. "-------- RESERVADO --------"
6. "Utilização de VAPA para função G92 [0=NO, 1=YES]"
[simple drill]
1. "Drill/Counterbore"
7. ""
8. ""
9. ""
10. ""
11. ""
[peck drill]
3. ""
7. "Peck"
8. ""
9. ""
10. ""
11. ""
[chip break]
```

```
3. ""
7. "Peck"
8. ""
9. ""
10. ""
11. ""
[tap]
3. ""
7. ""
8. ""
9. ""
10. ""
11. ""
[bore1]
1. "Bore #1 (feed-out)"
7. ""
8. ""
9. ""
10. ""
11. ""
[bore2]
1. "Bore #2 (stop spindle, rapid out)"
3. ""
7. ""
8. ""
9. ""
10. ""
11. ""
[misc1]
1. "Fine Bore (shift)"
7. ""
8. ""
9. ""
10. ""
[misc2]
1. "Rigid Tapping Cycle"
3. ""
7. ""
8. ""
9. ""
10. ""
11. ""
[drill cycle descriptions]
7. "Fine bore (shift)"
8. "Rigid Tapping Cycle"
[canned text]
1. "Stop"
2. "Ostop"
3. "Bld on"
4. "bLd off"
5. "M5"
6. "M6"
7. "M7"
8. "M8"
9. "M9"
10. "M10"
[CTRL_TEXT_END]
```